A ZDD-based solver for combinatorial reconfiguration problems

Jun Kawahara Kyoto University

Joint work with

Takehiro Ito, Yu Nakahata, Takehide Soh, Akira Suzuki, Junichi Teruyama, Takahisa Toda

Self introduction

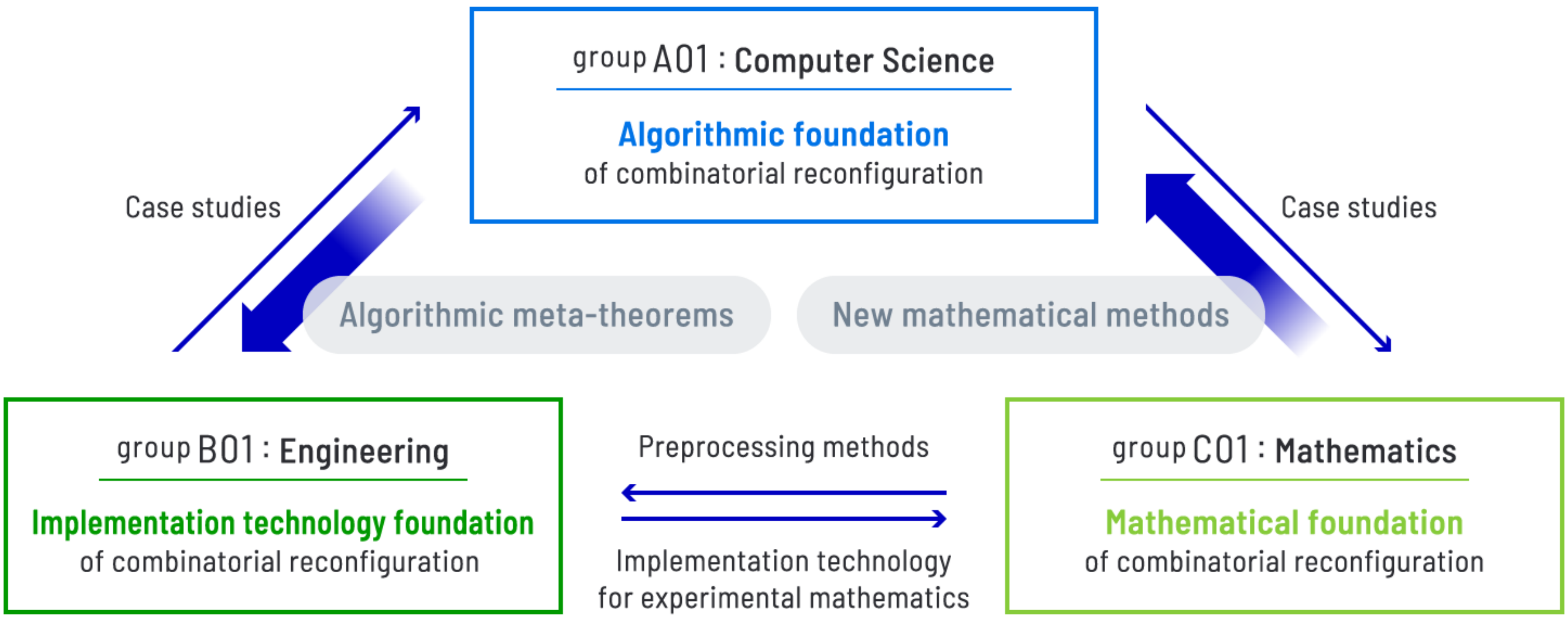
- Jun KAWAHARA
  - In 2000-2009, student at Kyoto University (supervisor: Kazuo Iwama)
  - In 2010-2012, researcher at JST ERATO Minato Discrete Structure Manipulation System Project (leader: Shin-ichi Minato)
  - In 2013-2019 , assistant professor at Nara Institute of Science and Technology (NAIST)
  - Currently, associate professor at Kyoto University
- My research topics:
  - Graph optimization/enumeration algorithms using zero-suppressed binary decision diagrams (ZDDs)

Project I join

Fusion of Computer Science, Engineering and Mathematics Approaches for Expanding Combinatorial Reconfiguration

Head Investigator: Takehiro Ito

Leader: Takehiro Ito



Leader: Jun Kawahara

Leader: Yoshio Okamoto

https://core.dais.is.tohoku.ac.jp/en/project/project_summary/

Members of group B01

Jun Kawahara	ZDD
Daisuke Iioka	Power engineering
Takahisa Toda	Model checking
Takehide Soh	SAT solver
Akira Suzuki	Reconfiguration
Junichi Teruyama	SAT complexity
Yu Nakahata	ZDD
Takehiro Ito (A01)	Reconfiguration

We are developing solvers based on several methods.

ZDD

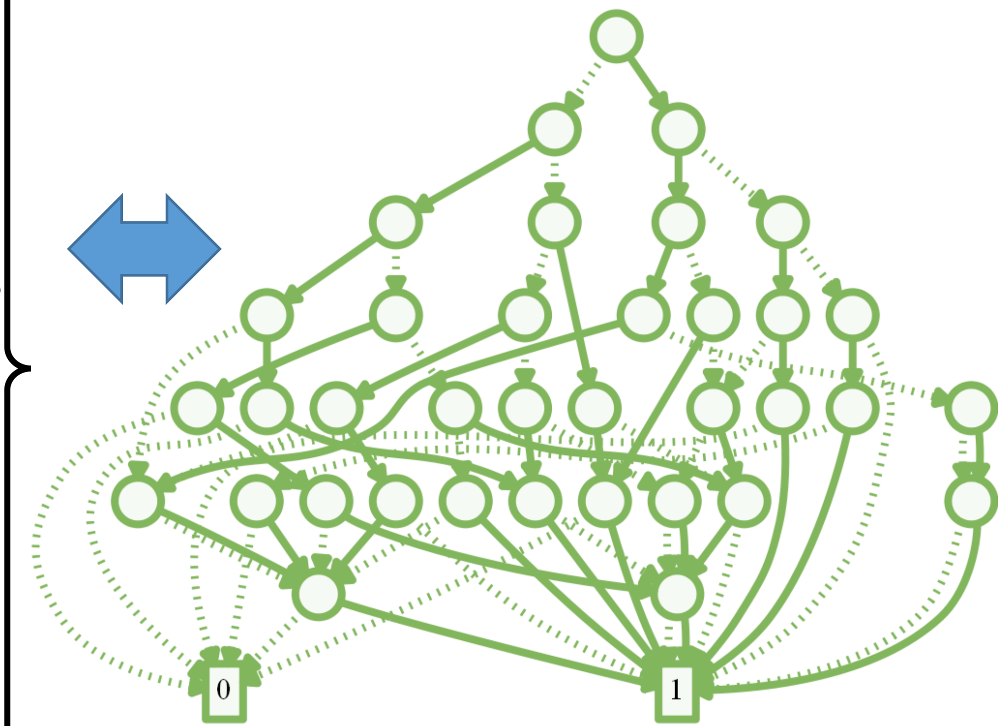
- Zero-suppressed Binary Decision Diagram
- Proposed by [Minato 1993]
- Compactly and efficiently stores a family of sets

family of sets

{2, 3, 5}, {1, 2, 3, 4}, {1, 3}, {3, 6}, {2, 5, 6, 7},
{1, 2, 6, 7}, {1, 6, 7}, {1, 2, 5, 7}, {2, 3, 6},
{2, 5, 6, 7}, {1, 2, 4, 5, 6, 7}, {1, 4}, {1, 5, 6},
{1, 2, 3, 5, 7}, {1, 2, 3, 6}, {1, 2}, {1, 6, 7},
{1, 2, 4, 7}, {2, 5, 6, 7}, {1, 3, 4, 5, 6}, {1, 3},
{5, 6, 7}, {1, 4, 5, 6, 7}, {3, 6, 7}, {3, 4, 7}, {1},
{2}, {6, 7}, {1, 2, 5}, {7}, {2, 5, 7}, {2, 6},
{1, 5, 7}, {3, 5, 7}, {1, 2, 6, 7}, {2, 3, 5, 6, 7},
{2, 5}, {2, 3, 4, 6}, {}, {2, 3}, {1, 6}, {1, 2, 4},
{2, 3, 5, 7}, {2, 3, 6, 7}, {3, 5, 6, 7}, {1, 5, 6},
{3}, {2, 6, 7}, {3, 4}, {2, 4, 6, 7}, {1, 2, 3, 4},
{2, 3, 5}, {1, 2, 3, 6, 7}, {1, 2, 3, 4, 6}, {5, 7},
{5}, {2, 5, 6, 7}, {1, 3, 4, 6}, {1, 2, 5, 6},
{2, 3, 4, 5, 6}, {3, 4, 5, 6}, {3, 4, 7}, {1, 5, 7},
{3, 4, 5, 7}

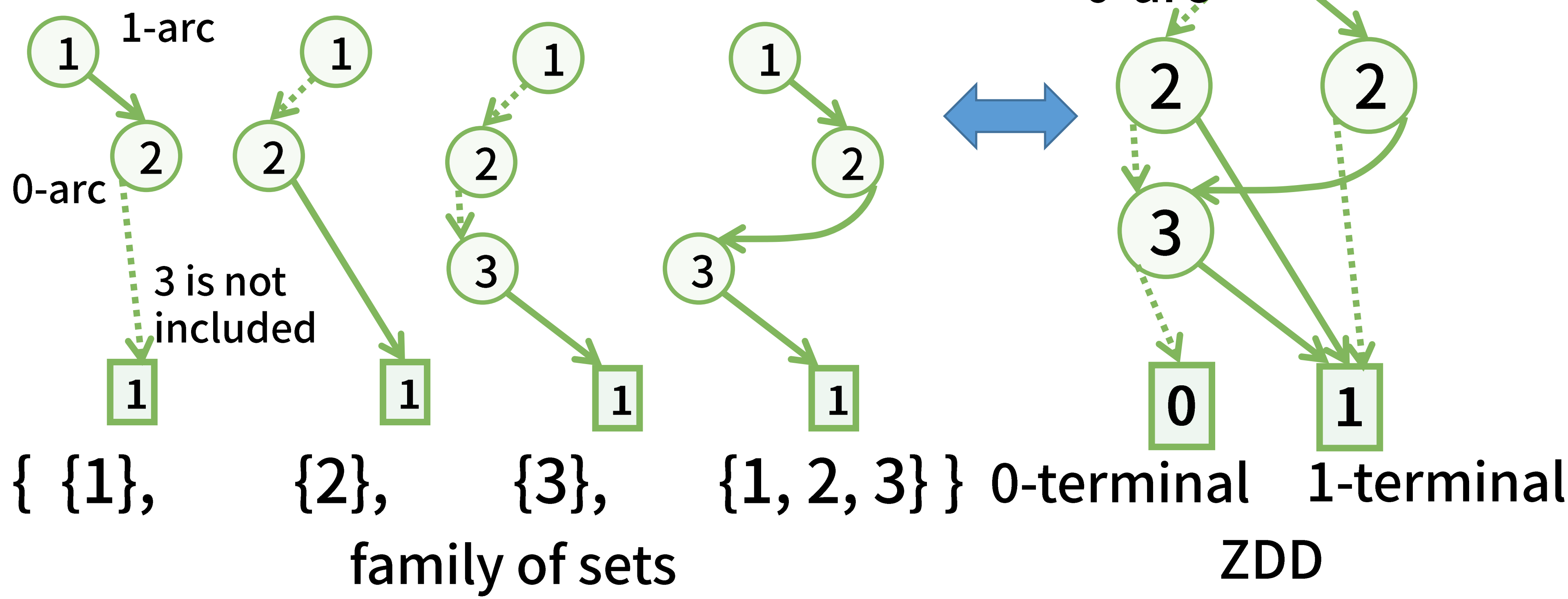
(directed acyclic graph)

ZDD



How to read ZDD

One to one correspondence between a set and a path from the root to 1

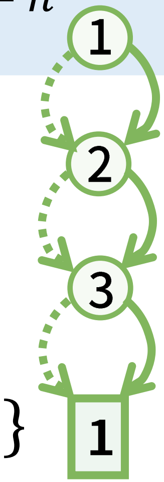


: node with label i

$i < j$ if i points at j

Features of ZDDs

size = n

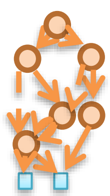


- The size (the number of non-terminal nodes) of a ZDD is sometimes exponentially smaller than the cardinality of the family the ZDD represents.

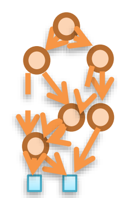
$\{ X \mid X \in 2^{\{1,\ldots,n\}} \}$

- Rich ZDD operations:
  - Extract the sets including (not including) a specified element

sets including 4



{1, 2, 5},
{1, 4},
{1, 3, 4, 7},
{3},
{4},...

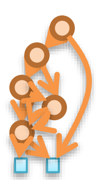


~~{1, 2, 5},~~
{1, 4},
{1, 3, 4, 7},
~~{3},~~
{4},...

Features of ZDDs

- The size (the number of non-terminal nodes) of a ZDD is sometimes exponentially smaller than the cardinality of the family the ZDD represents.
- Rich ZDD operations:
  - Extract the sets including (not including) a specified element
  - Set operations (called family algebra by Knuth) union, intersection, subtract, superset, subset,...

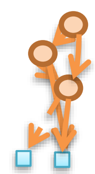
$\mathcal{F}$



{1, 2, 5},
{1, 3, 4, 7},...

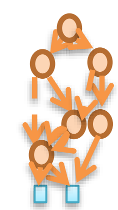
$\cup$

$\mathcal{G}$



{1, 4},
{1, 3, 4, 7} ,...

$\mathcal{F} \cup \mathcal{G}$



{1, 2, 5},
{1, 4},
{1, 3, 4, 7},...

union

$|\mathcal{F}|$: size of $\mathcal{F}$

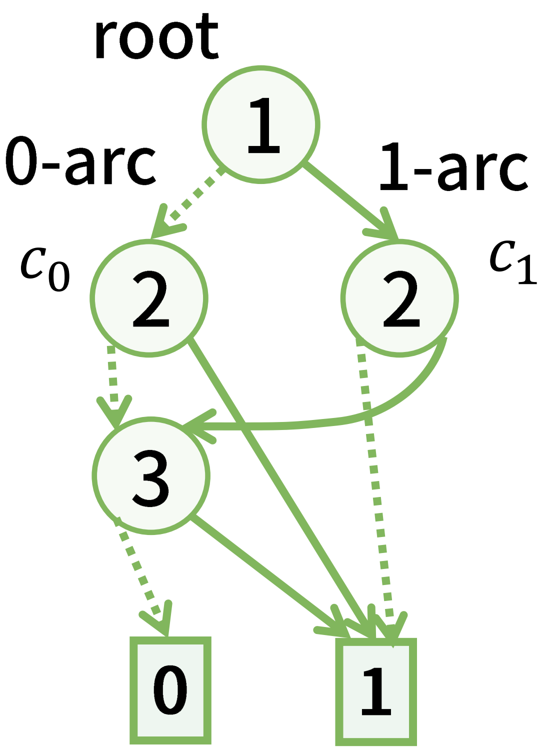
Time complexity: $\theta(|\mathcal{F}||\mathcal{G}|)$
practically, in many cases, in proportion to $|\mathcal{F}| + |\mathcal{G}|$

Features of ZDDs

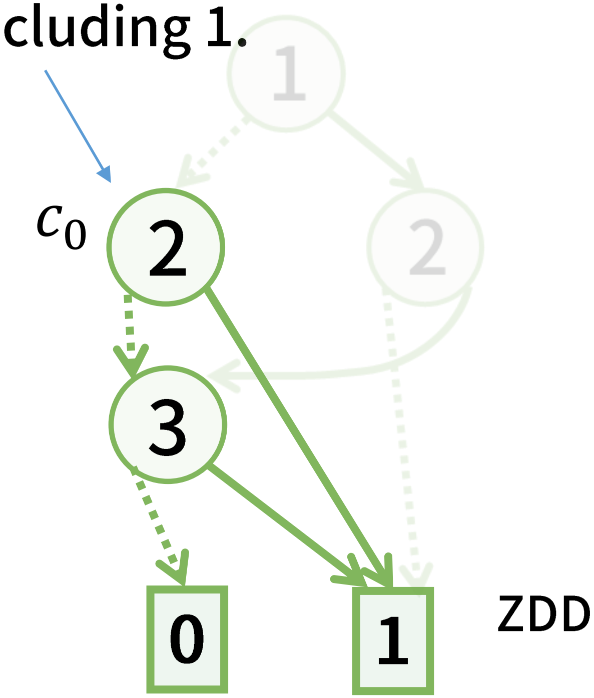
- The size (the number of non-terminal nodes) of a ZDD is sometimes exponentially smaller than the cardinality of the family the ZDD represents.
- Rich ZDD operations:
  - Extract the sets including (not including) a specified element
  - Set operations (called family algebra by Knuth) union, intersection, subtract, superset, subset,...
  - Count the number of sets in the family
  - Uniformly random sampling
  - Obtain the K lightest/heaviest sets

Recursive structure of ZDD

- Let c_i be the node pointed at by i-arc of the root.
- We can regard nodes reachable from c_i as a ZDD.



This ZDD represents the family of sets not including 1.



not including 1

{ {2}, {3},

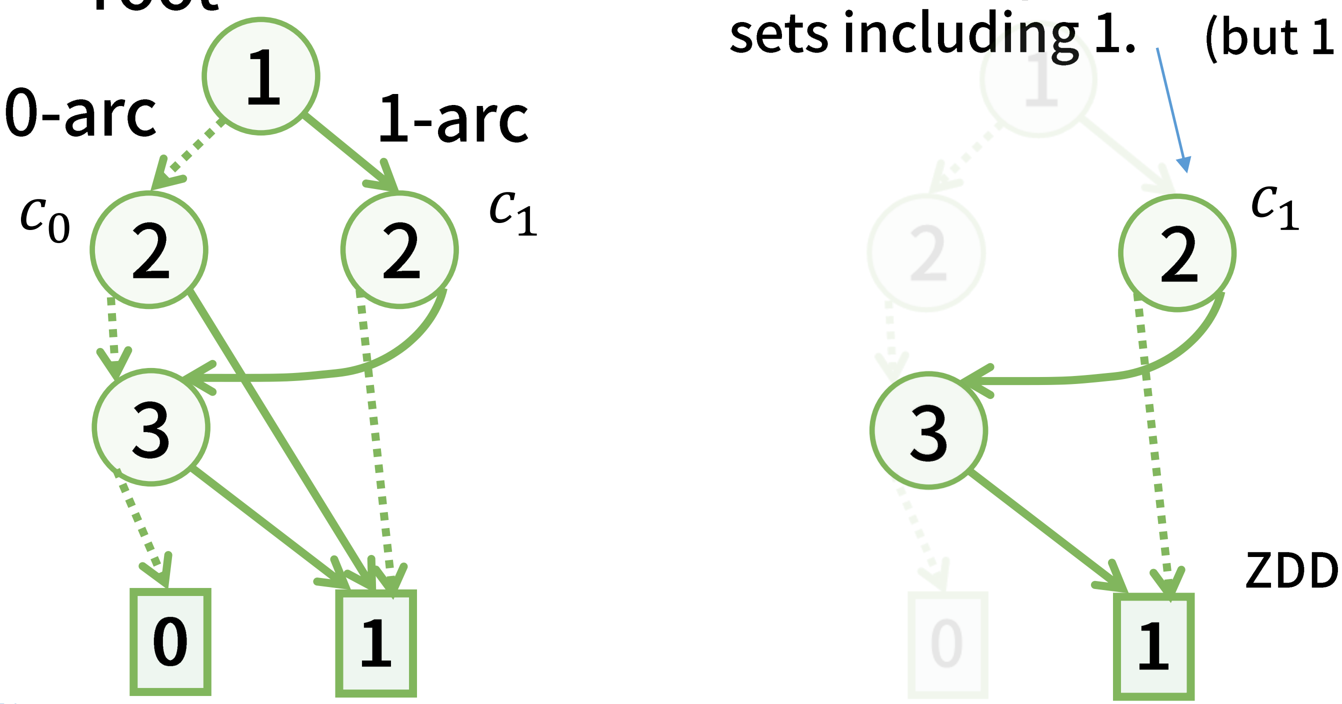
including 1 {1}, {1, 2, 3} }

→ { {2}, {3} }

Recursive structure of ZDD

- Let c_i be the node pointed at by i-arc of the root.
- We can regard nodes reachable from c_i as a ZDD.

This ZDD represents the family of sets including 1. (but 1 is removed)



not including 1

{ {2}, {3},

including 1 {1}, {1, 2, 3} }

remove 1 from each set

→ { {}, {2, 3} }

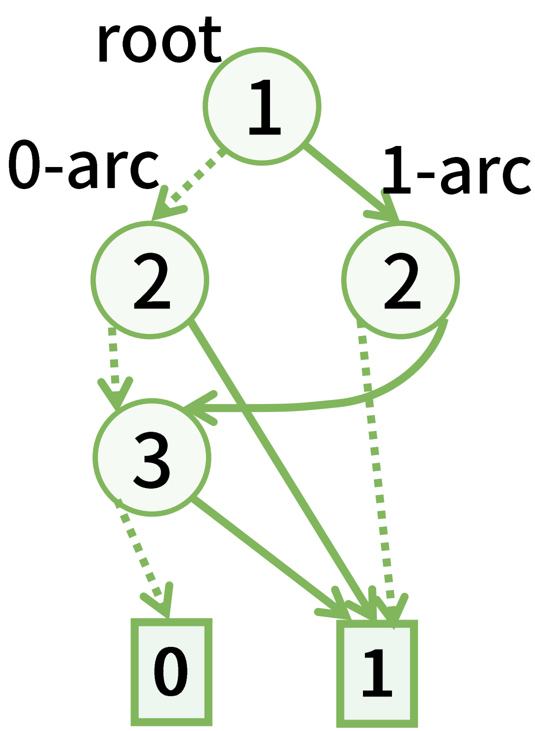
Recursive structure of ZDD

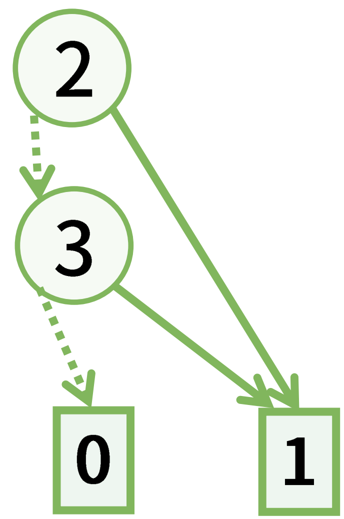
- We can decompose the family into two families of sets not including 1 and including 1.

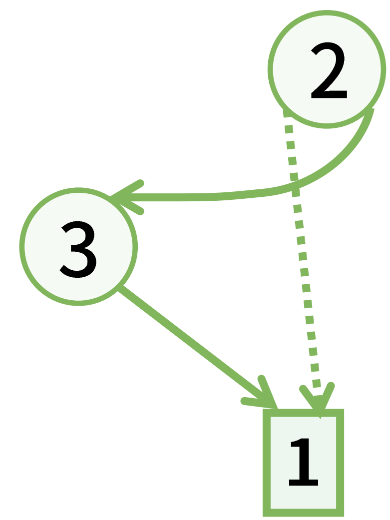
We introduce $\sqcup$ (join) operation

$$\mathcal{A} \sqcup \mathcal{B} = \{\, A \cup B \mid A \in \mathcal{A}, B \in \mathcal{B} \,\}$$

$$\{\{1\}\} \times \{\, \{\}, \{2, 3\} \,\} = \{\, \{1\}, \{1, 2, 3\} \,\}$$



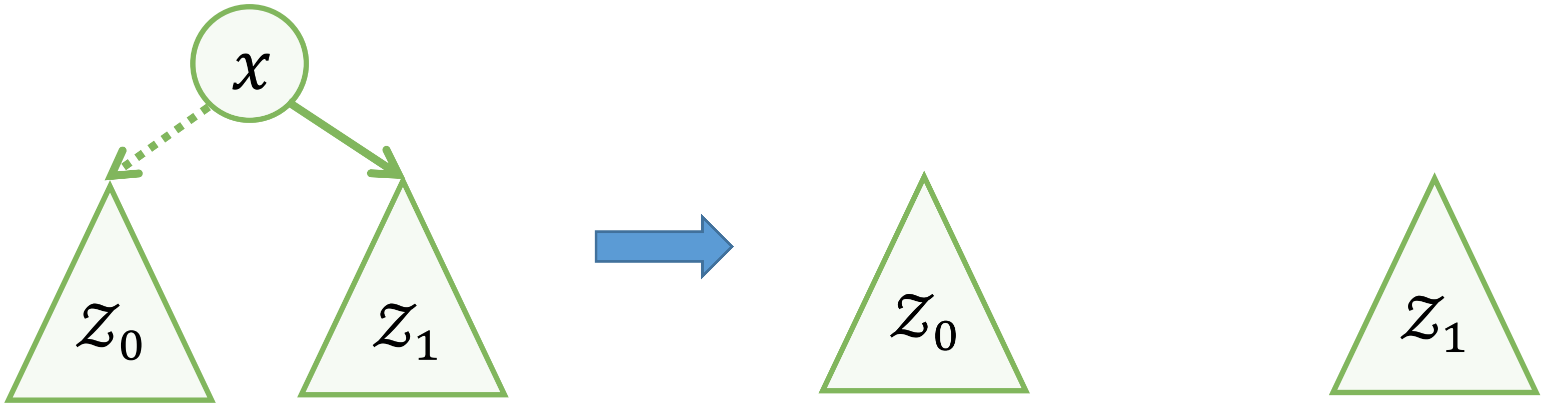




$$\{\, \{1\}, \{2\}, \{3\}, \{1, 2, 3\} \,\} \;=\; \{\, \{2\}, \{3\} \,\} \;\cup\; \{\{1\}\} \sqcup \{\, \{\}, \{2, 3\} \,\}$$

Recursive structure of ZDD

- In general,



ZDD $\mathcal{Z}$ representing family $\mathcal{F}$

ZDD $\mathcal{Z}_0$ representing family $\mathcal{F}_0$

ZDD $\mathcal{Z}_1$ representing family $\mathcal{F}_1$

$$\mathcal{F} = \mathcal{F}_0 \cup \{\{x\}\} \sqcup \mathcal{F}_1$$

$$\mathcal{F}_0 = \{\, F \mid F \in \mathcal{F}, x \notin F \,\}$$

$$\mathcal{F}_1 = \{\, F \setminus \{x\} \mid F \in \mathcal{F}, x \in F \,\}$$

Intersection operation of two ZDDs [Bryant 1986], [Minato 1993]

- Algorithm for computing the intersection of two families as ZDDs

$$\{\{1\}, \{2\}, \{3\}, \{1, 2, 3\}\} \cap \{\{1\}, \{1, 2\}, \{1, 2, 3\}\}$$
$$= \{\{1\}, \{1, 2, 3\}\}$$

Intersection operation of two ZDDs [Bryant 1986], [Minato 1993]

- Algorithm for computing the intersection of two families as ZDDs

$\{\{1\}, \{2\}, \{3\}, \{1, 2, 3\}\} \cap \{\{1\}, \{1, 2\}, \{1, 2, 3\}\}$

$= \{\{1\}, \{1, 2, 3\}\}$

Idea: use the recursive structure of ZDDs

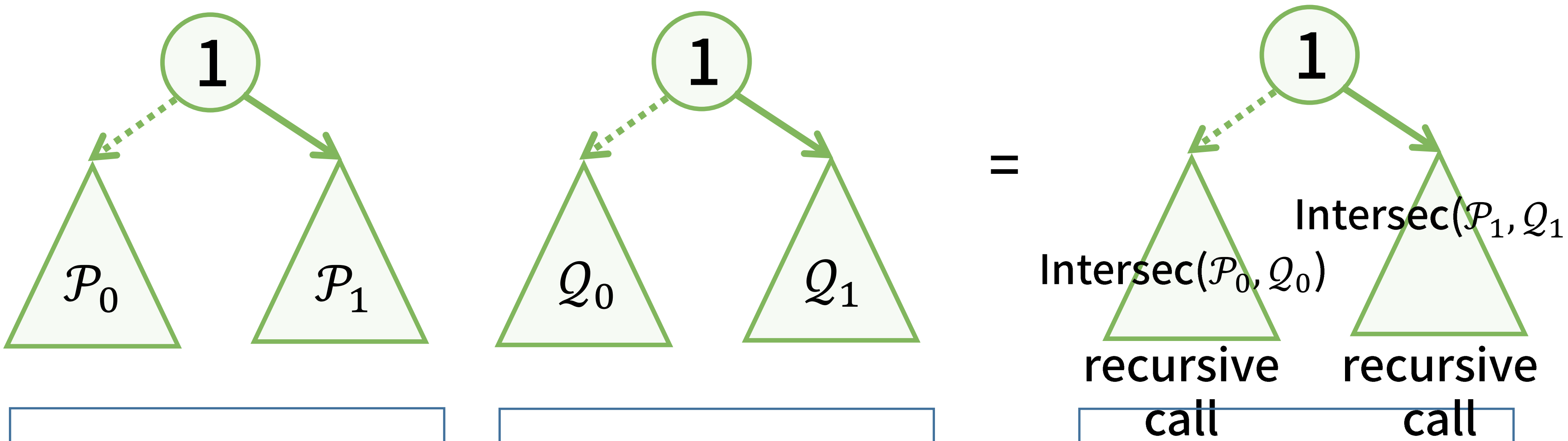
Let $\mathrm{Intersec}(\mathcal{P}, \mathcal{Q}) = \mathcal{P} \cap \mathcal{Q}$.

$\mathcal{P}$	$\mathcal{Q}$	$\mathrm{Intersec}(\mathcal{P}, \mathcal{Q})$
$\mathcal{P}_0$ not including 1	$\mathcal{Q}_0$ not including 1	$\mathrm{Intersec}(\mathcal{P}_0, \mathcal{Q}_0)$ not including 1
$\{\{1\}\} \sqcup \mathcal{P}_1$ including 1	$\{\{1\}\} \sqcup \mathcal{Q}_1$ including 1	$\{\{1\}\} \sqcup \mathrm{Intersec}(\mathcal{P}_1, \mathcal{Q}_1)$ including 1

Intersection operation of two ZDDs [Bryant 1986], [Minato 1993]

- Algorithm for computing the intersection

We use the same symbol for a family and its ZDD as $\mathcal{P}, \mathcal{Q}$.



$\mathcal{P}$	$\mathcal{Q}$	Intersec($\mathcal{P}, \mathcal{Q}$)
$\mathcal{P}_0$ not including 1	$\mathcal{Q}_0$ not including 1	Intersec($\mathcal{P}_0, \mathcal{Q}_0$) not including 1
$\{\{1\}\} \sqcup \mathcal{P}_1$ including 1	$\{\{1\}\} \sqcup \mathcal{Q}_1$ including 1	$\{\{1\}\} \sqcup$ Intersec($\mathcal{P}_1, \mathcal{Q}_1$) including 1

Intersection operation of two ZDDs [Bryant 1986], [Minato 1993]

- **Algorithm** $\mathrm{Intersec}(\mathcal{P}, \mathcal{Q})$

0 ∅ **1** {∅}
family consisting only of emptyset

If $\mathcal{P}$ is **0** or $\mathcal{Q}$ is **0** , return **0**.
0-terminal

If $\mathcal{P}$ is **1** and $\mathcal{Q}$ is **1** , return **1**.
1-terminal

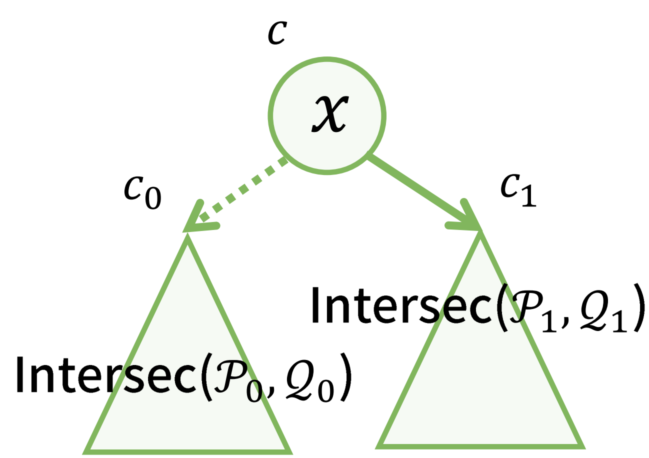
Assume that the labels of roots of $\mathcal{P}$ and $\mathcal{Q}$ are x.

(We omit the other cases)

$c_0 \leftarrow \mathrm{Intersec}(\mathcal{P}_0, \mathcal{Q}_0)$

$c_1 \leftarrow \mathrm{Intersec}(\mathcal{P}_1, \mathcal{Q}_1)$

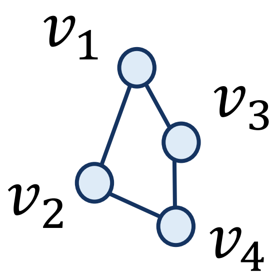
Create node c with label x.
Make i-arc of c point at c_i.
Return c.



Construction of a ZDD for independent sets

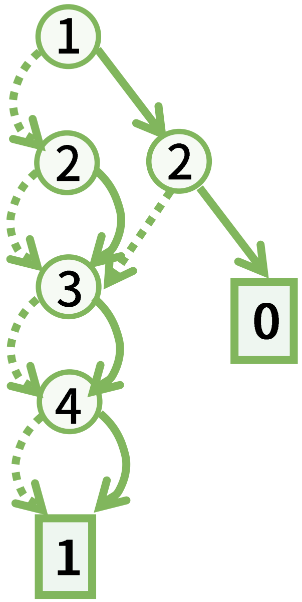
See e.g. [Knuth 2011].

- Given a graph $G = (V, E)$, we can construct a ZDD representing all the independent sets of G.



all independent sets

$\{\emptyset,\ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_1, v_4\}, \{v_2, v_3\}\}$



$\overline{\mathcal{X}_{1,2}}$

all the sets including at most one of i and j

$$\overline{\mathcal{X}_{i,j}} := \{\, X \mid X \in 2^{\{1,\ldots,n\}}, i \notin X \text{ or } j \notin X \}$$

By just computing

$$\bigcap_{\{u,v\} \in E} \overline{\mathcal{X}}_{u,v}$$

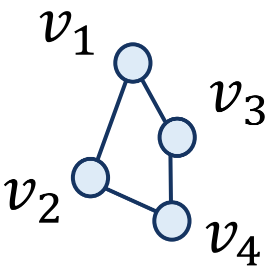
we obtain the ZDD.

Once the ZDD is constructed, we can easily enumerate all the elements.

Construction of a ZDD for independent sets

See e.g. [Knuth 2011].

- Given a graph $G = (V, E)$, we can construct a ZDD representing all the independent sets of G.



all independent sets

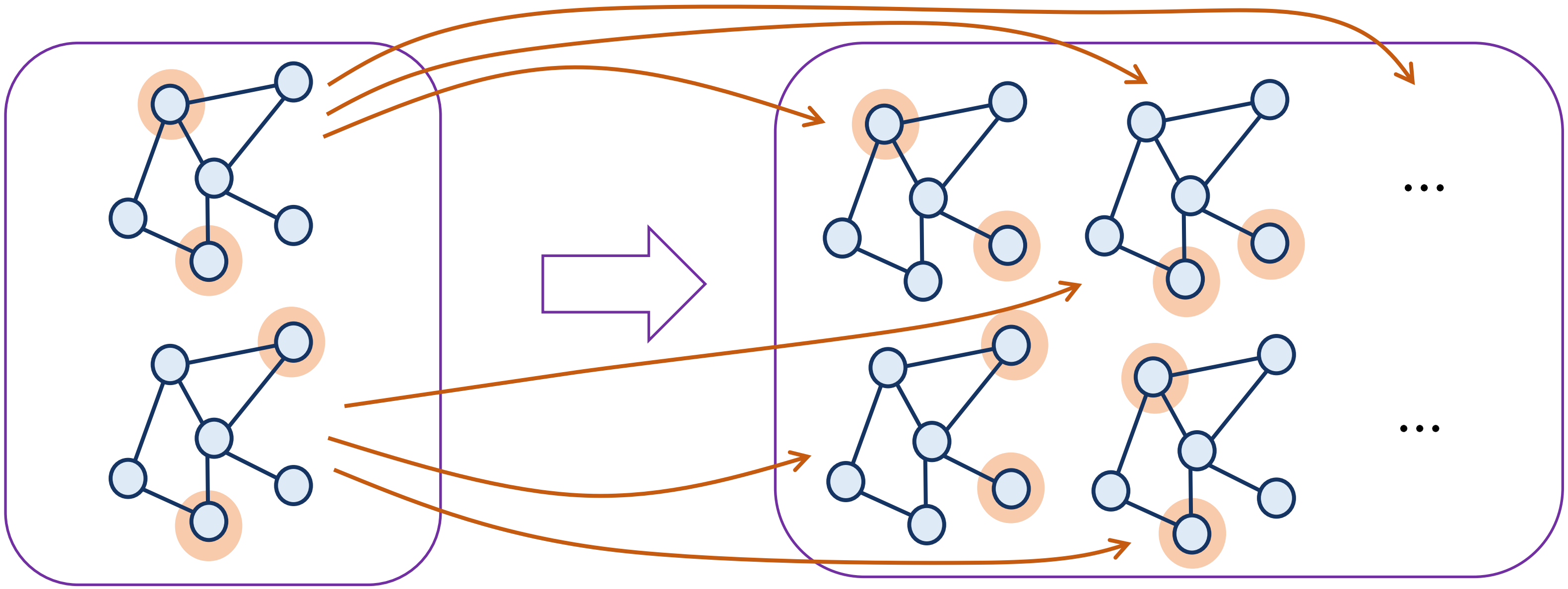
$\{\emptyset,\ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_1, v_4\}, \{v_2, v_3\}\}$

[Hayase-Sadakane-Tani 1995] designed a more efficient algorithm for constructing the ZDD (omitted).

Enumeration to reconfiguration

- Since we have all the independent sets as a ZDD, we expect that it can be used for solving reconfiguration problems.

Reconfiguration using ZDD

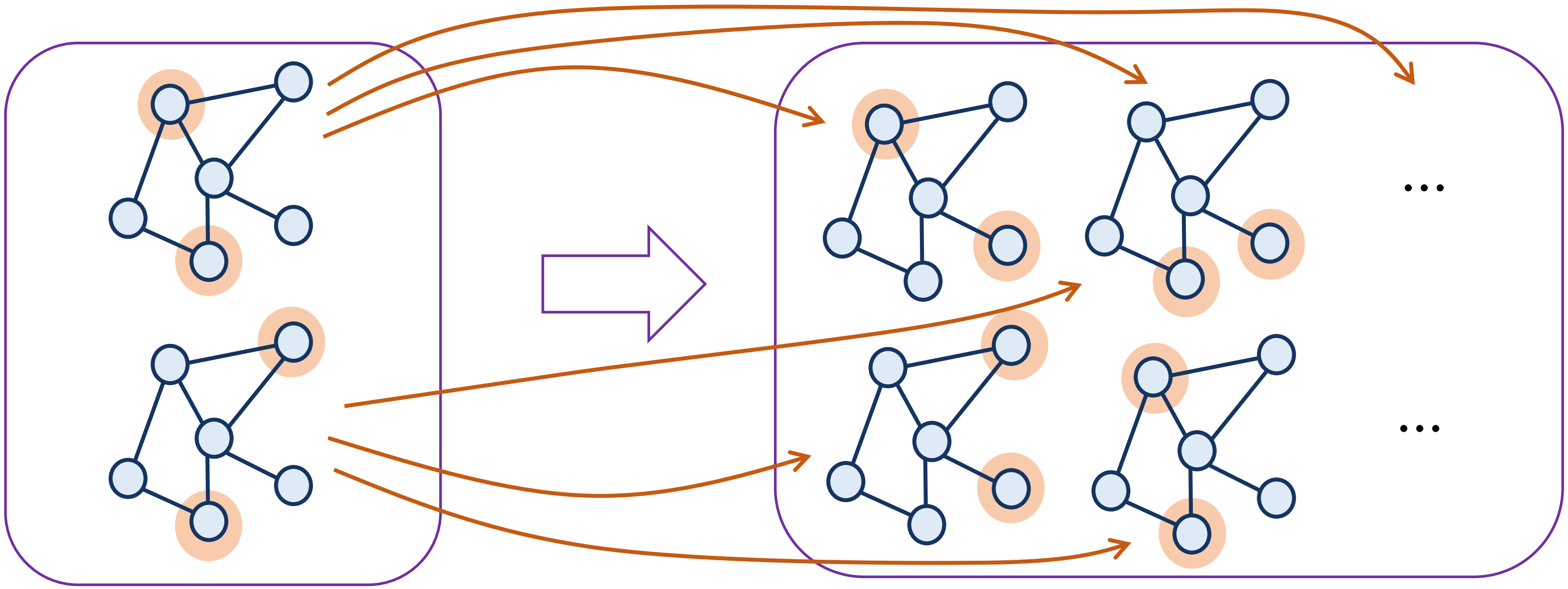


independent sets represented as a ZDD

independent sets obtained by one step of TJ (token jumping)

We want to construct a ZDD representing it.

Reconfiguration using ZDD



independent sets represented as a ZDD

$\mathcal{F}$

independent sets obtained by one step of TJ (token jumping)

$\mathrm{swap}(\mathcal{F}, V) \cap \mathcal{F}_{\mathrm{sol}}$

We define

$$\mathrm{swap}(\mathcal{F}, A) := \{\ F \cup \{v\} \setminus \{v'\} \mid F \in \mathcal{F}, v \notin F, v \in A, v' \in F\ \}.$$

but, $\mathrm{swap}(\mathcal{F}, A)$ includes not independent sets.

Let $\mathcal{F}_{\mathrm{sol}}$ be the family of all the independent sets of G.

Our result

published only in a domestic conference

- Given a family $\mathcal{F}$ of sets as a ZDD, we design algorithms for constructing the following ZDDs.

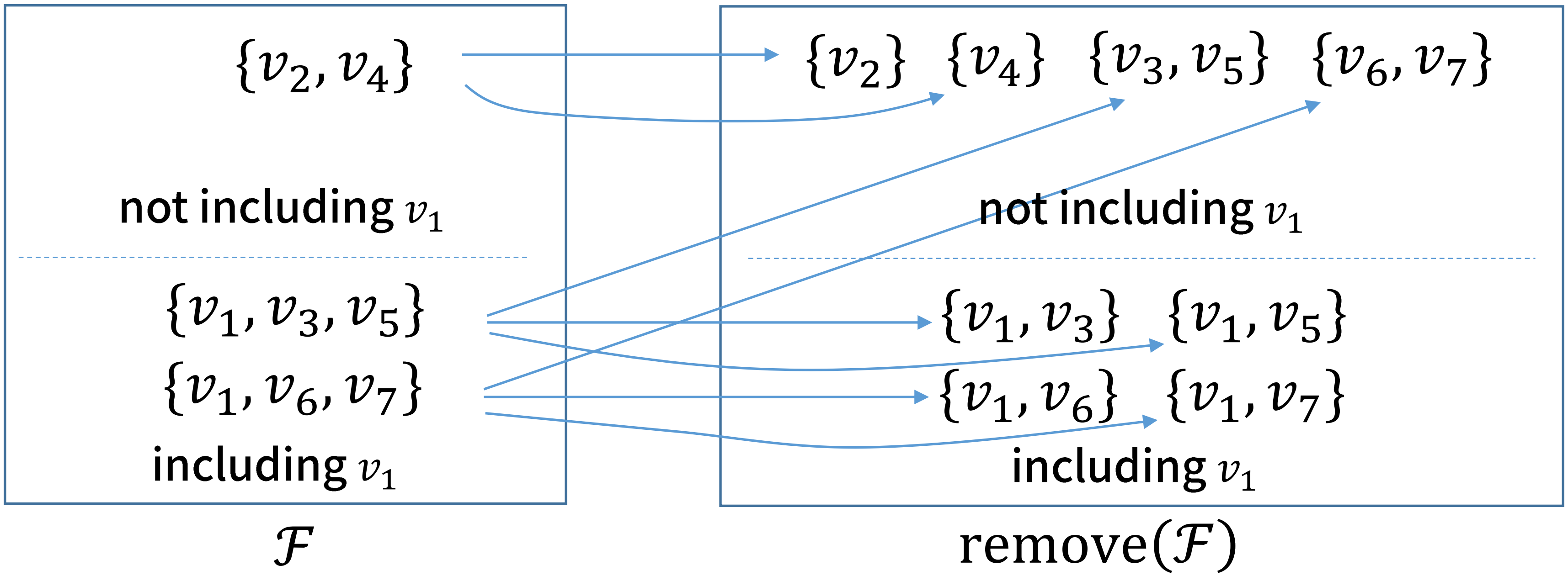
$$\mathrm{swap}(\mathcal{F}, A) = \{\, F \cup \{v\} \setminus \{v'\} \mid F \in \mathcal{F}, v \notin F, v \in A, v' \in F\}$$

$$\mathrm{remove}(\mathcal{F}) = \{\, F \setminus \{v\} \mid F \in \mathcal{F}, v \in F \,\}$$

$$\mathrm{add}(\mathcal{F}, A) = \{\, F \cup \{v\} \mid F \in \mathcal{F}, v \notin F, v \in A\}$$

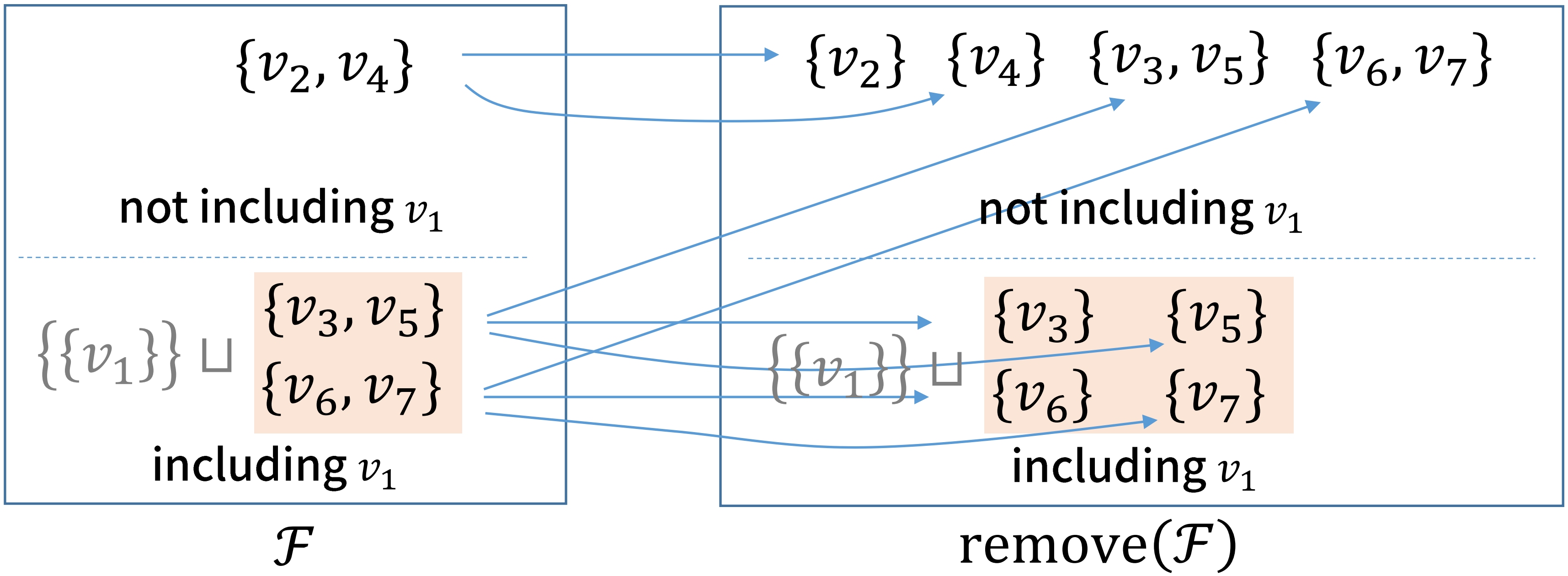
How to construct a ZDD for remove

$$\text{remove}(\mathcal{F}) = \{\, F \setminus \{v\} \mid F \in \mathcal{F}, v \in F \,\}$$



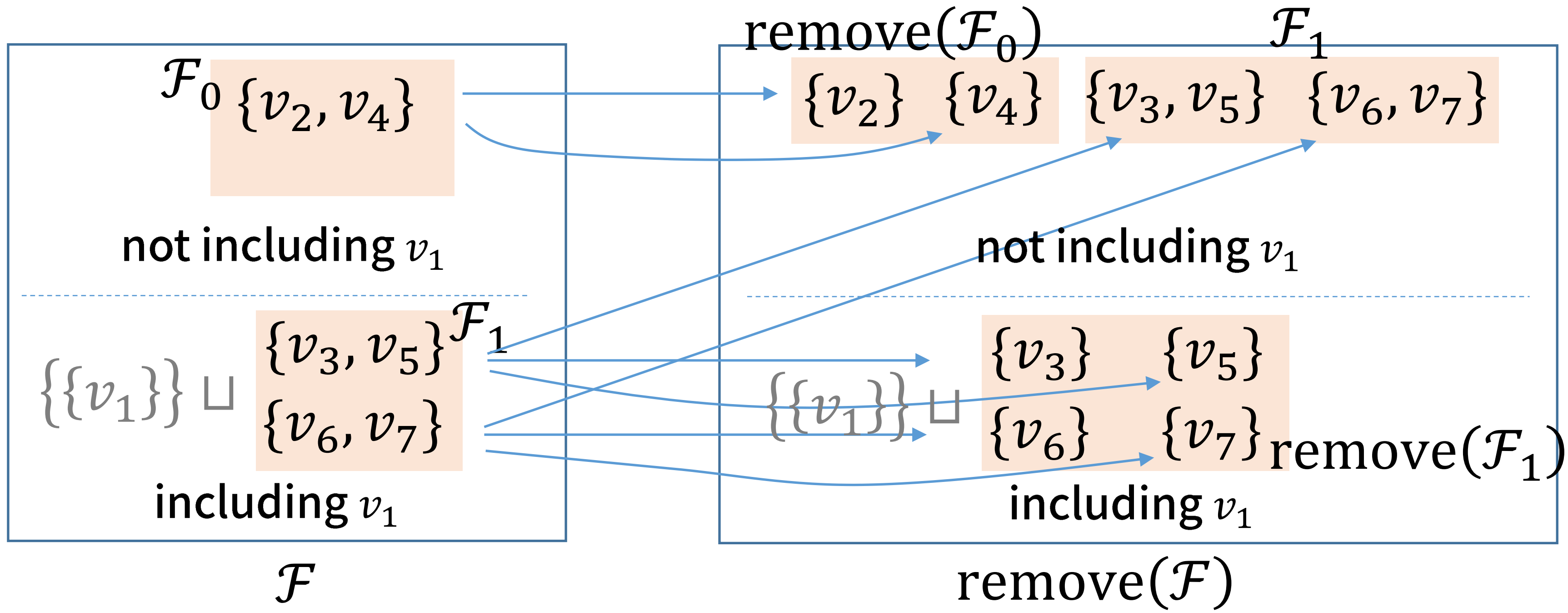
How to construct a ZDD for remove

$$\mathrm{remove}(\mathcal{F}) = \{\, F \setminus \{v\} \mid F \in \mathcal{F}, v \in F \,\}$$



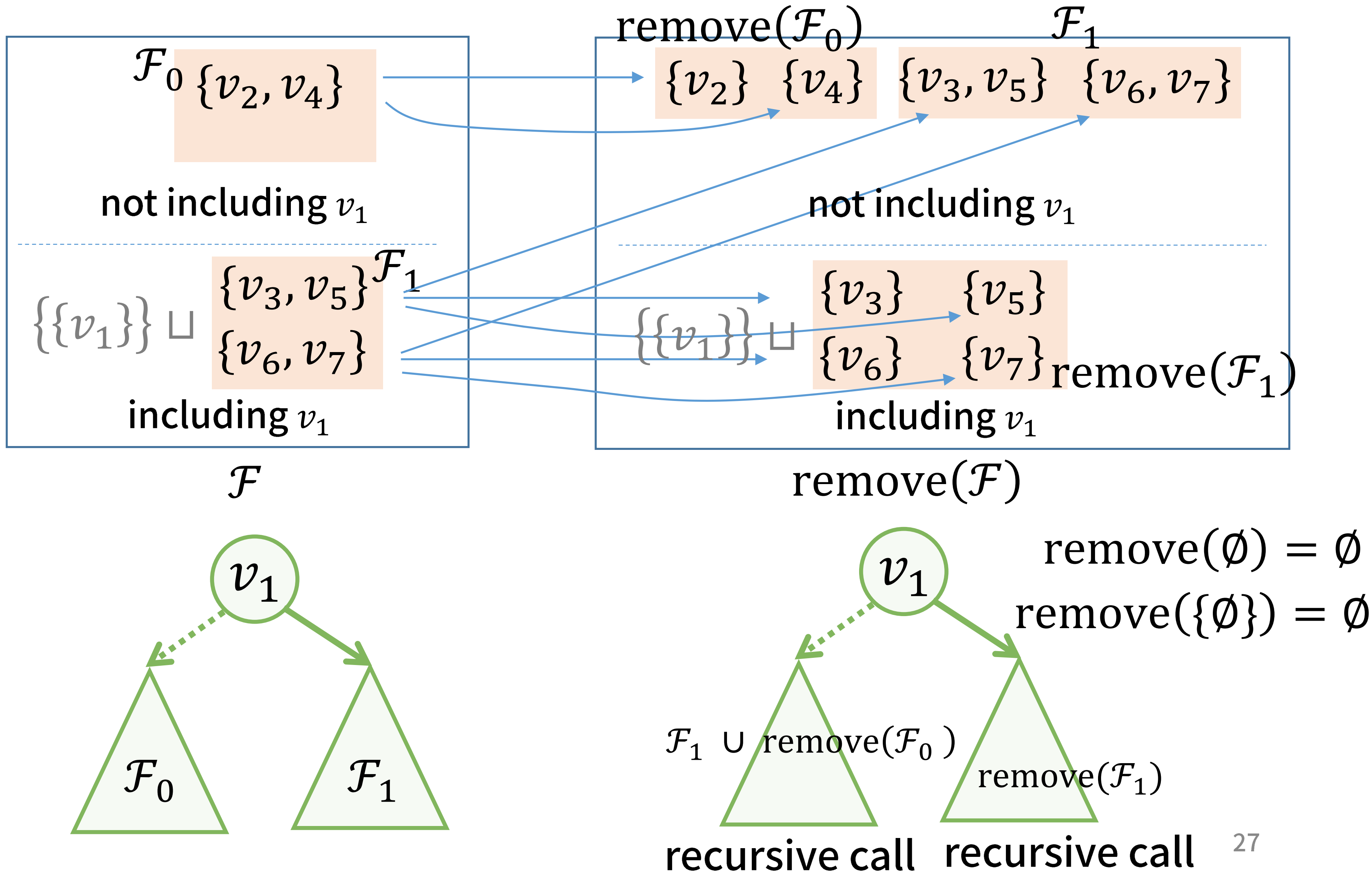
How to construct a ZDD for remove

$\mathrm{remove}(\mathcal{F}) = \{\, F \setminus \{v\} \mid F \in \mathcal{F}, v \in F \,\}$



How to construct a ZDD for remove

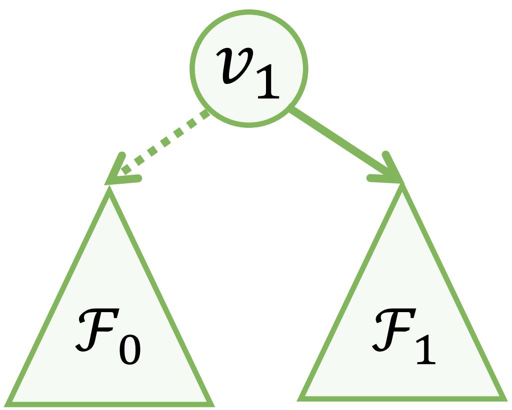
$$\mathrm{remove}(\mathcal{F}) = \{ F \setminus \{v\} \mid F \in \mathcal{F}, v \in F \}$$

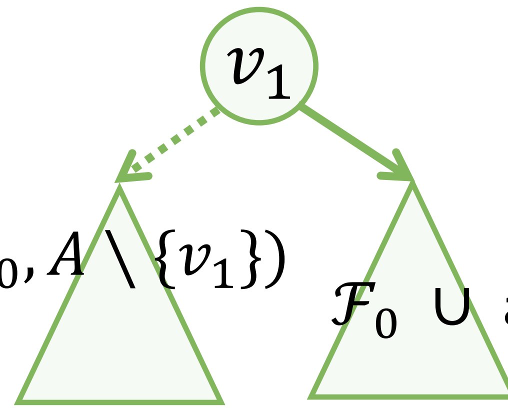


How to construct a ZDD for add

$$\mathrm{add}(\mathcal{F}, A) = \{\, F \cup \{v\} \mid F \in \mathcal{F}, v \notin F, v \in A \}$$

Consider only the case where v_1 is the smallest element in A.





$\mathrm{add}(\mathcal{F}_0, A \setminus \{v_1\})$

$\mathcal{F}_0 \cup \mathrm{add}(\mathcal{F}_1, A \setminus \{v_1\})$

recursive call recursive call

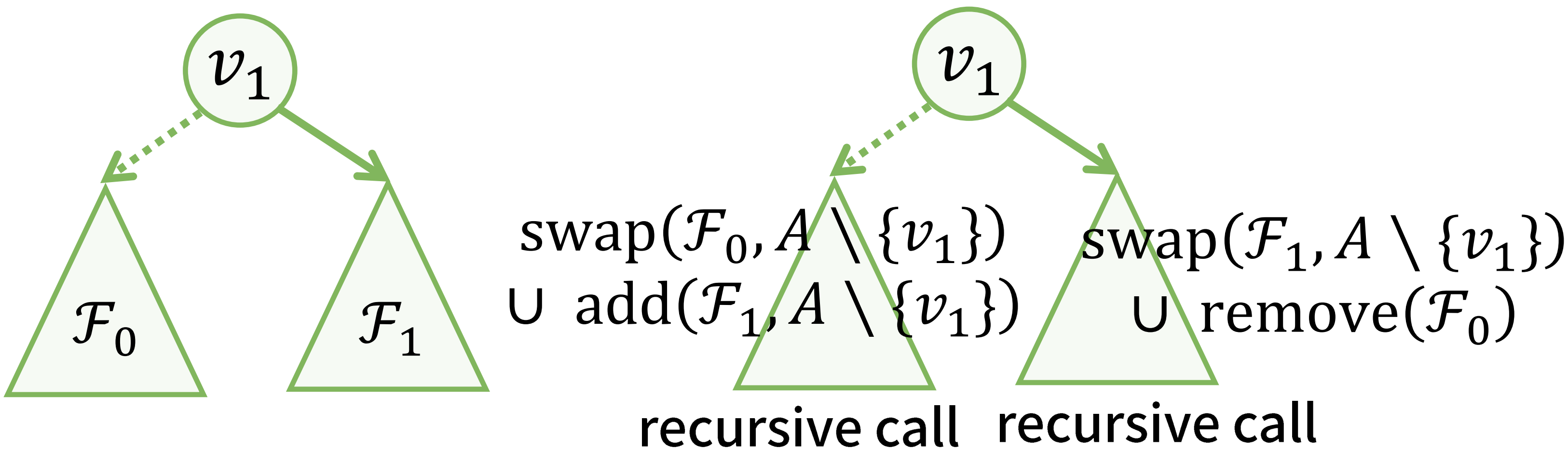
$$\mathrm{add}(\emptyset, A) = \emptyset$$

$$\mathrm{add}(\{\emptyset\}, A) = \{\{v\} \mid v \in A\}$$

How to construct a ZDD for swap

$$\mathrm{swap}(\mathcal{F}, A) = \{\, F \cup \{v\} \setminus \{v'\} \mid F \in \mathcal{F}, v \notin F, v \in A, v' \in F \}$$

Consider only the case where v_1 is the smallest element in A.



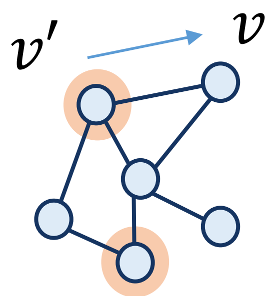
$$\mathrm{swap}(\emptyset, A) = \emptyset$$

$$\mathrm{swap}(\{\emptyset\}, A) = \emptyset$$

Slide operation for TS (token sliding)

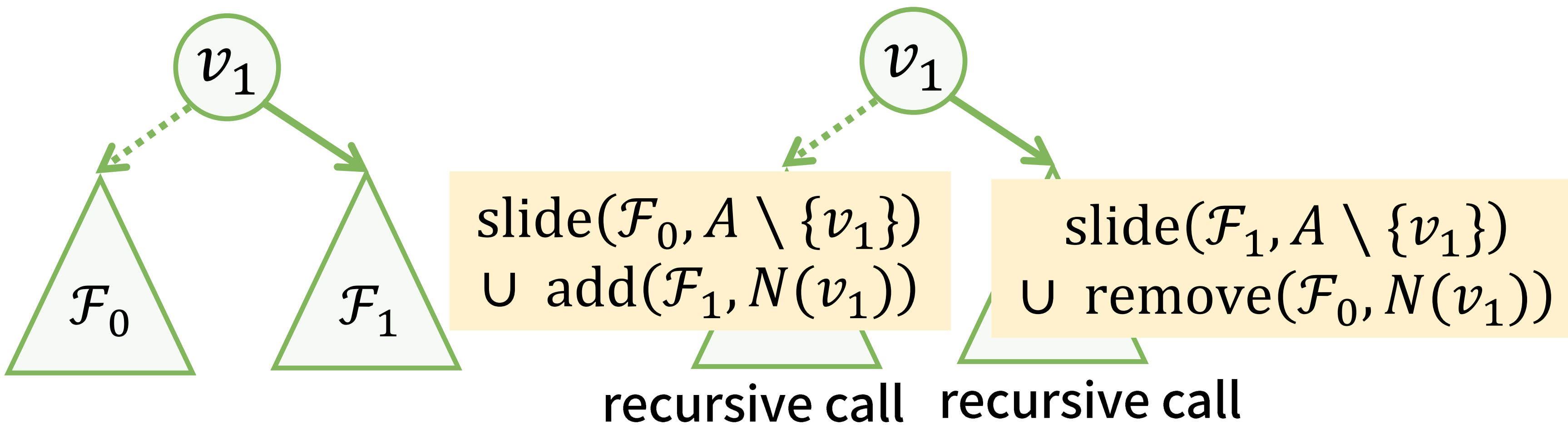
$\mathrm{slide}(\mathcal{F}, A) = \{\, F \cup \{v\} \setminus \{v'\} \mid F \in \mathcal{F}, v \notin F, v \in A, v' \in F,$

$\{v, v'\} \in E \,\}$



$N(v)$: the set of neighbors of v

Consider only the case where v_1 is the smallest element in A.



Algorithm for the independent set reconfiguration problem

breadth-first search

- Let $\mathcal{F}_{\mathrm{sol}}$ be the set of all the independent sets of G.
- Let S and T be an initial and goal sets.
- $\mathcal{F}_0 \leftarrow \{ S \},\ i \leftarrow 1$
- $\mathcal{F}_i \leftarrow \mathrm{swap}(\mathcal{F}_{i-1}, V) \cap \mathcal{F}_{\mathrm{sol}} \setminus \mathcal{F}_{i-2}$

all the sets obtained by one step

extract only independent sets

remove past sets (for efficiency)

- If $\mathcal{F}_i$ is empty, output "No reconf sequence from S to T"
- If $T \in \mathcal{F}_i$, output "There is a reconf sequence from S to T with length i."
- $i \leftarrow i + 1$, and continue.

Experimental results

Surfnet graph in the Internet topology zoo [Knight+ 2011]
independent set reconfiguration, TJ,
The initial/goal sets are randomly generated.

$|V| = 50, \qquad |E| = 68$

the number of ZDD nodes of $\mathcal{F}_i$

```
Step  1  time = 0.000093, size =  117
Step  2  time = 0.000383, size =  391
Step  3  time = 0.001711, size =  984
Step  4  time = 0.005881, size = 1981
Step  5  time = 0.016275, size = 3298
Step  6  time = 0.034983, size = 4785
Step  7  time = 0.066430, size = 6150
Step  8  time = 0.124186, size = 7184
Step  9  time = 0.207905, size = 7757
Step 10  time = 0.294560, size = 7735
Step 11  time = 0.345743, size = 7097
Step 12  time = 0.294848, size = 5921
Step 13  time = 0.180891, size = 4461
Step 14  time = 0.074440, size = 2987
Step 15  time = 0.023286, size = 1731
```

We found the shortest
(15 step) reconf sequence.

Intel(R) Xeon(R) Gold 5215L CPU @ 2.50GHz, memory 1.5TB

Experimental results: unsolved

Columbus graph in the Internet topology zoo [Knight+ 2011]
independent set reconfiguration, TJ,
The initial/goal sets are randomly generated.

$|V| = 70, \qquad |E| = 85$

the number of ZDD nodes of $\mathcal{F}_i$

```
Step 1  time =   0.000210,  size =     201
Step 2  time =   0.001617,  size =    1292
Step 3  time =   0.013077,  size =    5234
Step 4  time =   0.160867,  size =   16242
Step 5  time =   0.456166,  size =   42007
Step 6  time =   2.421557,  size =   95135
Step 7  time =   9.316833,  size =  192958
Step 8  time =  28.091186,  size =  356404
Step 9  time =  69.303806,  size =  606360
Step 10 time = 184.181911,  size =  958185
Step 11 time = 306.821899,  size = 1413044
Step 12 time = 607.407846,  size = 1949045
Step 13 time = 773.472284,  size = 2517807
...
```

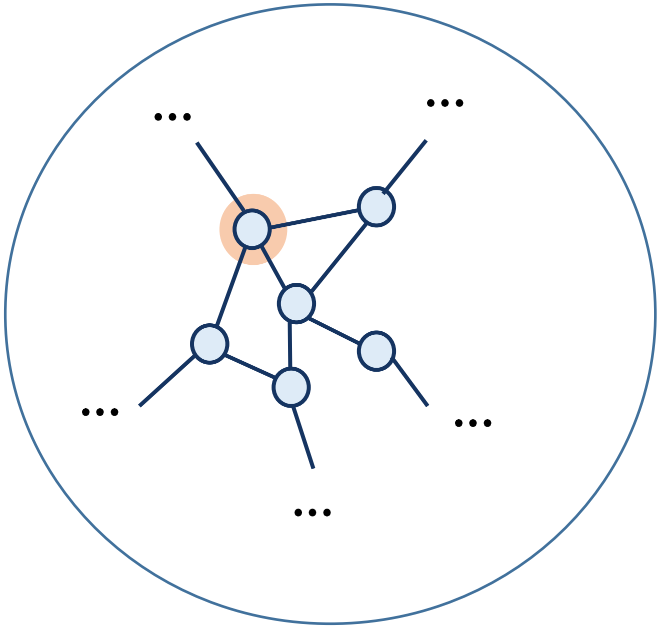
growing

Unfortunately, the ZDD-based solver sometimes cannot solve instances with only 70 vertices.

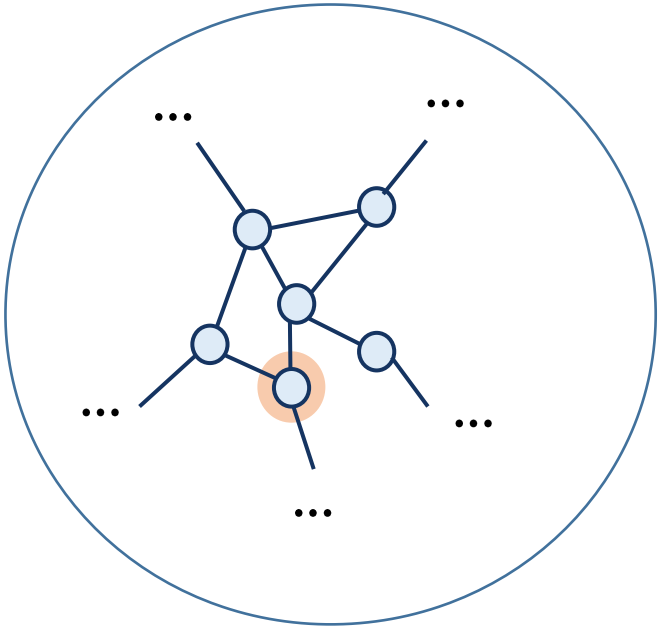
`Intel(R) Xeon(R) Gold 5215L CPU @ 2.50GHz, memory 1.5TB`

Unsolved instances

- The ZDD-based solver cannot solve instances with many vertices but having a trivial solution.



very large graph but the initial set consisting of one vertex



goal set

The ZDD-based solver first construct the ZDD representing all the independent sets, which consumes a lot of memory/time.

Experimental results

Instances used for proving PSPACE-completeness of the shortest path reconfiguration problem (translated to the independent set problem) in [Kamiński-Medvedev-Milanič 2011].

K	$\|V\|$	$\|E\|$	time [s]	Reconf sequence
9	117	738	15.58	5621
10	130	822	40.66	11253
11	143	906	109.82	22517
12	156	990	291.01	45045
13	169	1074	750.74	90101
14	182	1158	1874.71	180213
15	195	1242	4605.52	360437
16	208	1326	11584.92	720885
17	221	1410	28253.64	1441781
18	234	1494	70350.16	2883573
19	247	1578	175842.90	5767157

The sizes of intermediate ZDDs are at most 15,000.

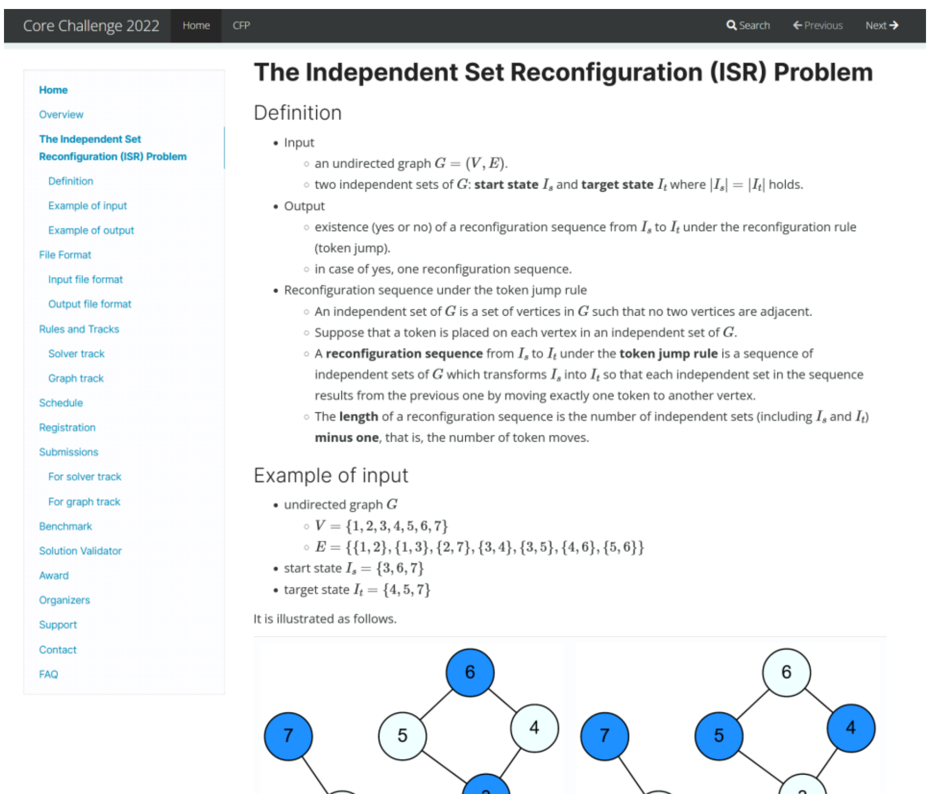
The number of candidates is small but the length of the sequence is very long, so breadth-search is efficient.

The ZDD-based solver excels such a type of instances.

Intel(R) Xeon(R) Gold 5215L CPU @ 2.50GHz, memory 1.5TB

Core Challenge

- Our project is organizing a programming competition, called “Core Challenge.”
- The 1st challenge is the independent set reconfiguration problem.


Core Challenge 2022 | Home | CFP | Search | Previous | Next

Home
Overview
The Independent Set Reconfiguration (ISR) Problem
Definition
Example of input
Example of output
File Format
Input file format
Output file format
Rules and Tracks
Solver track
Graph track
Schedule
Registration
Submissions
For solver track
For graph track
Benchmark
Solution Validator
Award
Organizers
Support
Contact
FAQ

The Independent Set Reconfiguration (ISR) Problem

Definition

- Input
  - an undirected graph $G = (V, E)$.
  - two independent sets of G: **start state** I_s and **target state** I_t where $|I_s| = |I_t|$ holds.
- Output
  - existence (yes or no) of a reconfiguration sequence from I_s to I_t under the reconfiguration rule (token jump).
  - in case of yes, one reconfiguration sequence.
- Reconfiguration sequence under the token jump rule
  - An independent set of G is a set of vertices in G such that no two vertices are adjacent.
  - Suppose that a token is placed on each vertex in an independent set of G.
  - A **reconfiguration sequence** from I_s to I_t under the **token jump rule** is a sequence of independent sets of G which transforms I_s into I_t so that each independent set in the sequence results from the previous one by moving exactly one token to another vertex.
  - The **length** of a reconfiguration sequence is the number of independent sets (including I_s and I_t) **minus one**, that is, the number of token moves.

Example of input

- undirected graph G
  - $V = \{1, 2, 3, 4, 5, 6, 7\}$
  - $E = \{\{1,2\}, \{1,3\}, \{2,7\}, \{3,4\}, \{3,5\}, \{4,6\}, \{5,6\}\}$
- start state $I_s = \{3, 6, 7\}$
- target state $I_t = \{4, 5, 7\}$

It is illustrated as follows.

Submission closed

A ZDD-based solver has been submitted, and will be compared with other solvers.

The results are being compiled.
A ZDD-based solver is slower than others for many instances…?

Generalization of the algorithm

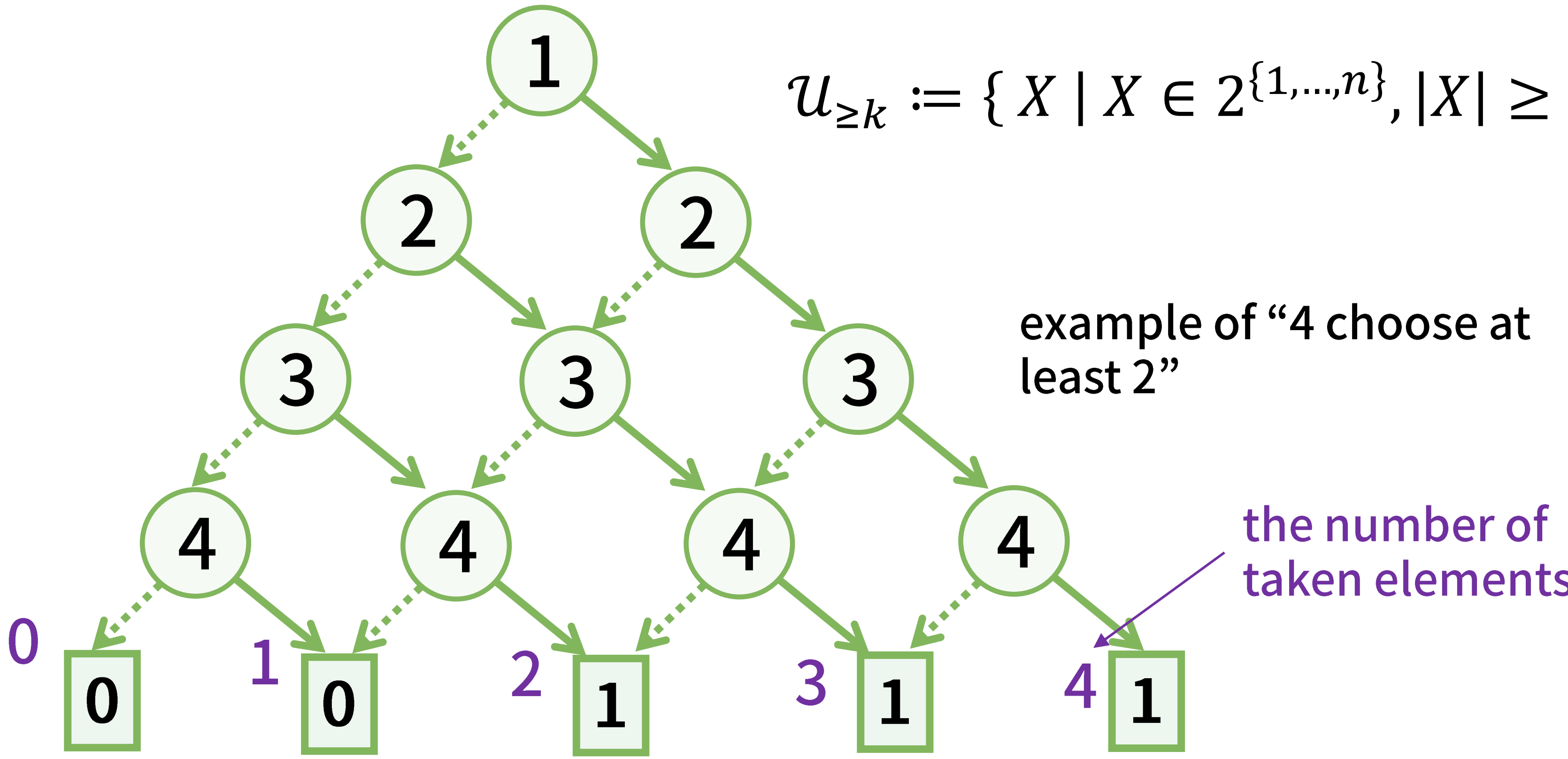
- Let $\mathcal{F}_{\mathrm{sol}}$ be the set of all the solutions.
- Let S and T be an initial and goal solutions.
- $\mathcal{F}_0 \leftarrow \{\, S \,\},\ i \leftarrow 1$
- $\mathcal{F}_i \leftarrow \mathrm{swap}(\mathcal{F}_{i-1}, V) \cap \mathcal{F}_{\mathrm{sol}} \setminus \mathcal{F}_{i-2}$

  - $\mathrm{swap}(\mathcal{F}_{i-1}, V)$: all the sets obtained by one step
  - $\mathcal{F}_{\mathrm{sol}}$: extract only solutions
  - $\mathcal{F}_{i-2}$: remove past sets (for efficiency)

- If $\mathcal{F}_i$ is empty, output "No reconf sequence from S to T"
- If $T \in \mathcal{F}_i$, output "There is a reconf sequence from S to T with length i."
- $i \leftarrow i + 1$, and continue.

TAR (token addition and removal)

- ZDD for "n choose at least k"



$$\mathcal{U}_{\geq k} \coloneqq \{ X \mid X \in 2^{\{1,\dots,n\}}, |X| \geq k \}$$

example of "4 choose at least 2"

Actually, the ZDD has exactly one 0- and 1- terminals.

By taking the intersection of $\mathcal{U}_{\geq k}$ and $\mathcal{F}_{\mathrm{sol}}$, we can impose the constraint that every feasible solution has at least k elements.

Using add and remove with the above solution space ZDD, we can solve the TAR model.

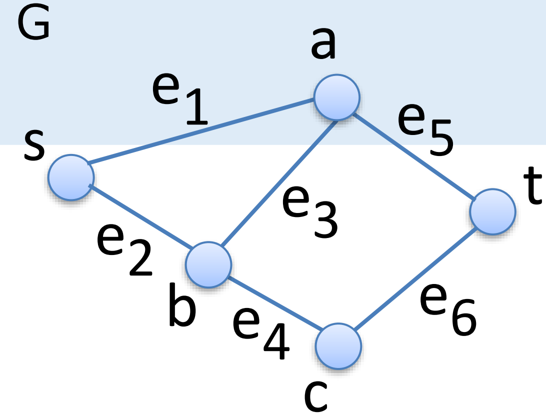
ZDDs we can construct

- We can construct ZDDs for the family of…
  - Independent sets
  - Cliques
  - Vertex covers
  - Dominating sets
  - Hitting sets [Knuth 2011]

Our algorithm can solve the reconfiguration versions of these problems by constructing ZDDs representing the family of the above target sets as the solution space ZDD.

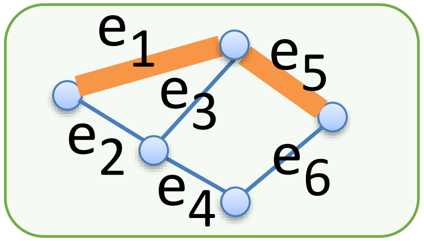
Set of subgraphs

input G

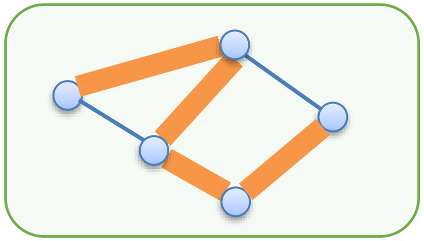


- Fixing an input graph G, we regard an edge set as a subgraph.

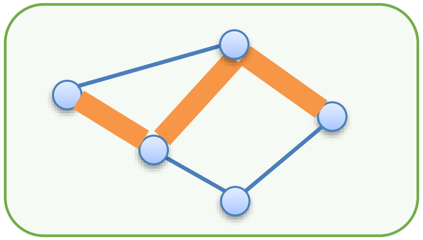
ex.) s-t paths



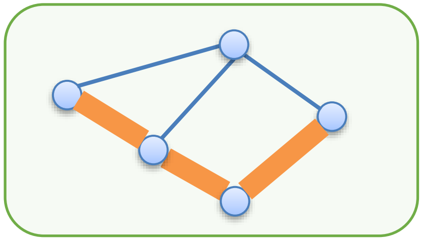
$\{e_1, e_5\}$



$\{e_1, e_3, e_4, e_6\}$



$\{e_2, e_3, e_5\}$

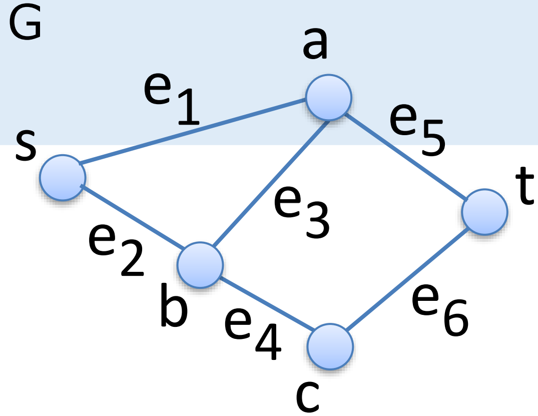


$\{e_2, e_4, e_6\}$

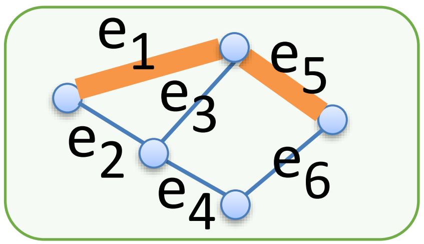
Set of subgraphs

- Fixing an input graph G, we regard an edge set as a subgraph.

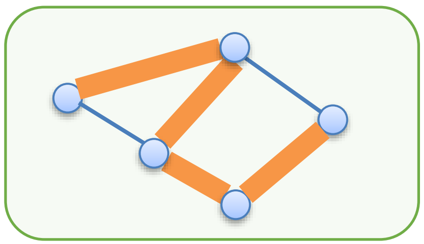
input G



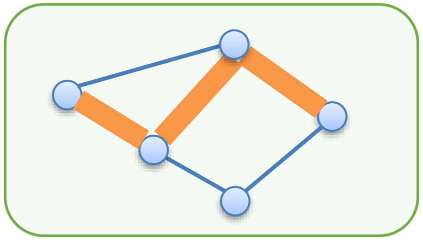
ex.) s-t paths



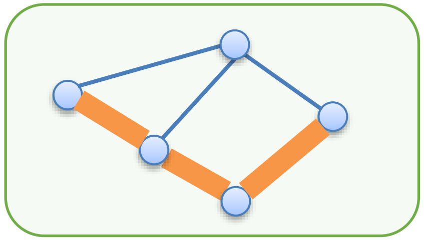
$\{e_1, e_5\}$



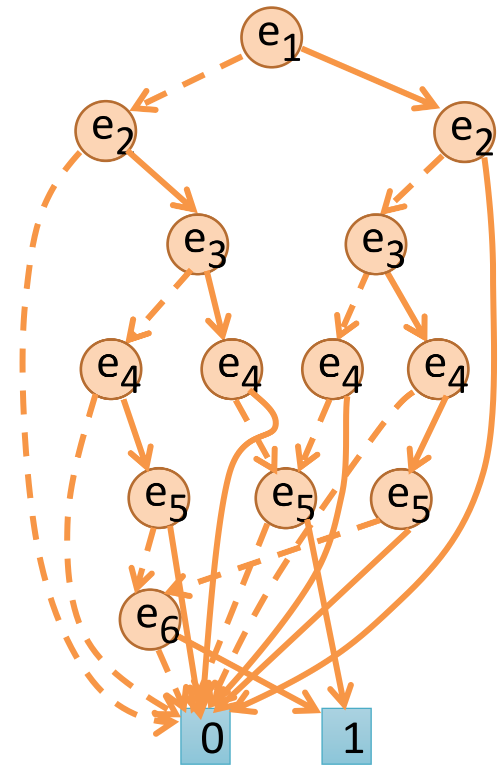
$\{e_1, e_3, e_4, e_6\}$



$\{e_2, e_3, e_5\}$



$\{e_2, e_4, e_6\}$

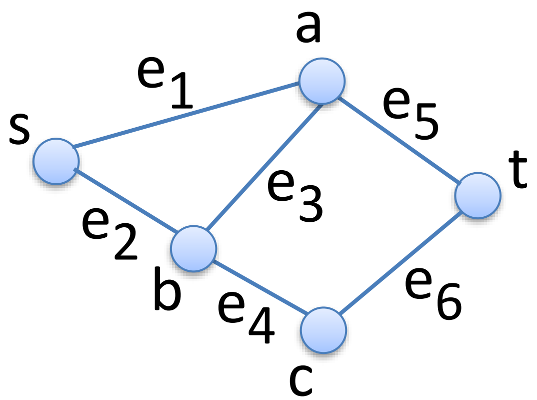


- A set of subgraphs can be represented by a ZDD.

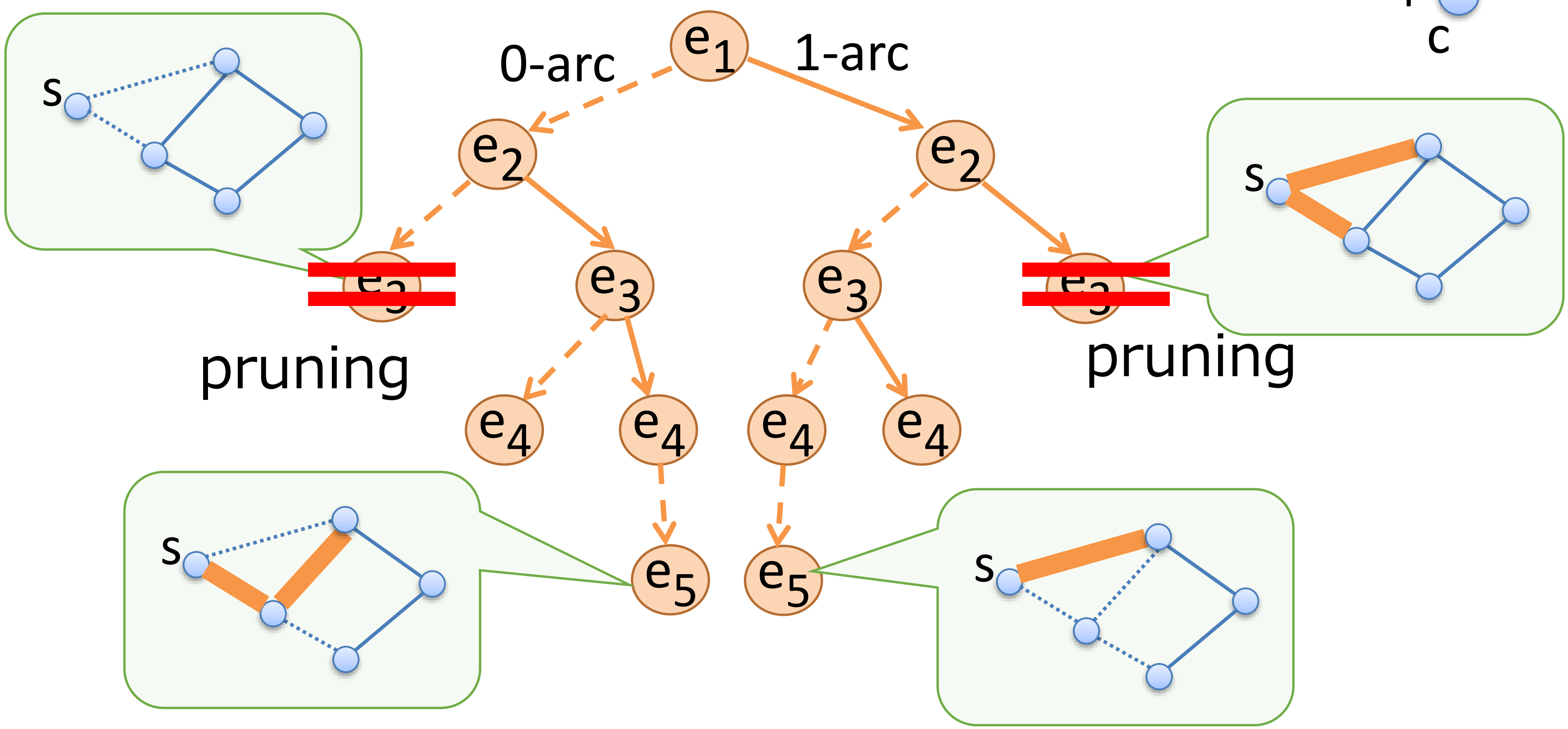
ZDD construction: frontier-based search (FBS)

[Sekine et al. 1995], [Knuth 2008], [Kawahara et al. 2017]

- constructs the ZDD representing a set of subgraphs (e.g., s-t paths)
  - in a top-down manner
  - by pruning and sharing nodes



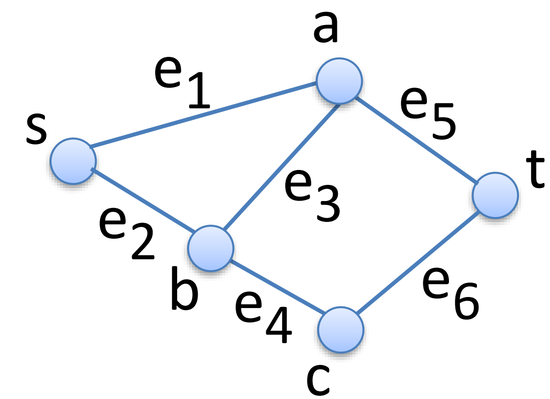
ex.) s-t paths



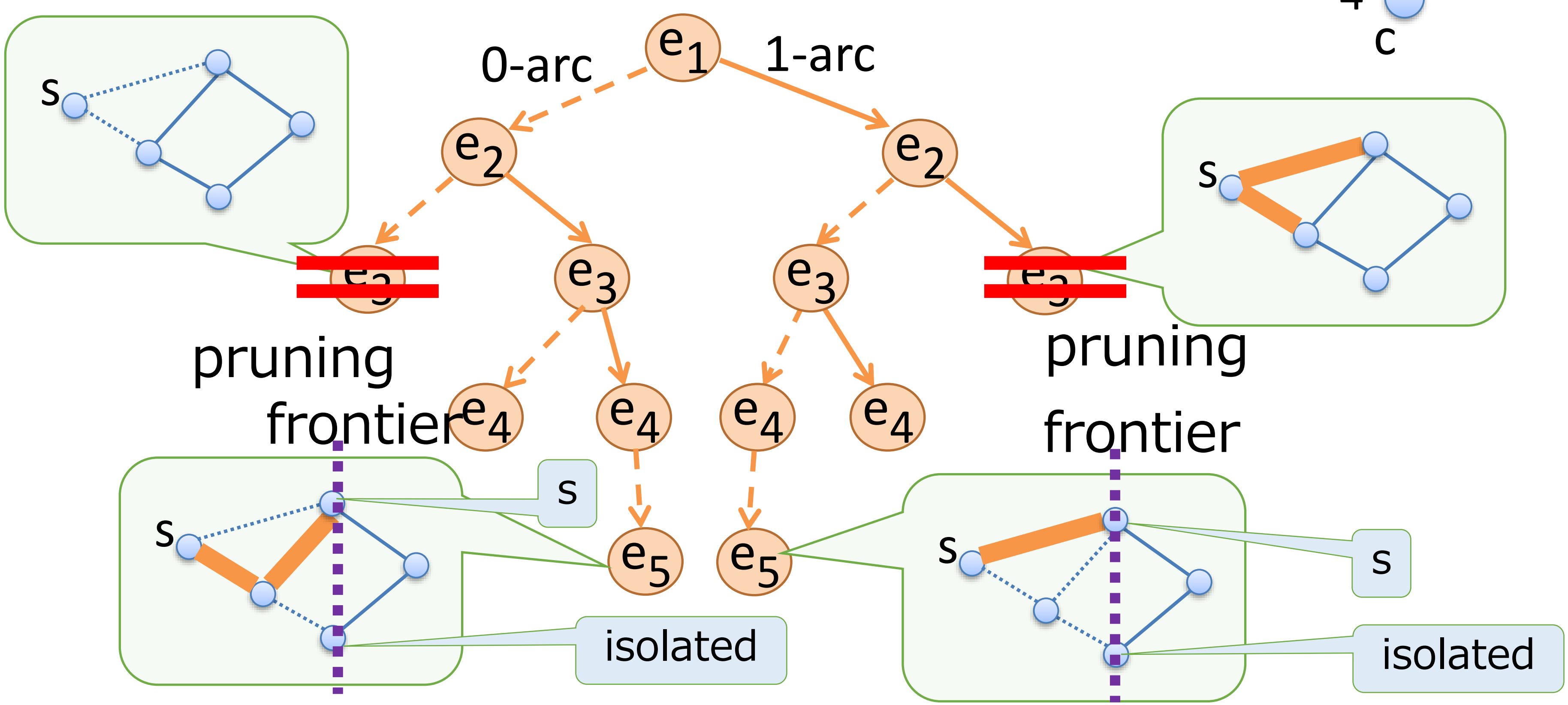
ZDD construction: frontier-based search (FBS)

[Sekine et al. 1995], [Knuth 2008], [Kawahara et al. 2017]

- constructs the ZDD representing a set of subgraphs (e.g., s-t paths)
  - in a top-down manner
  - by pruning and sharing nodes



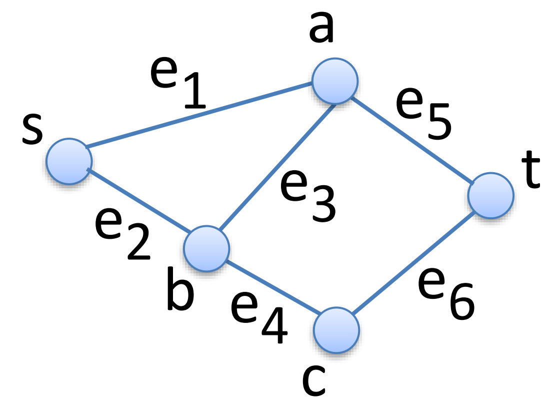
ex.) s-t paths



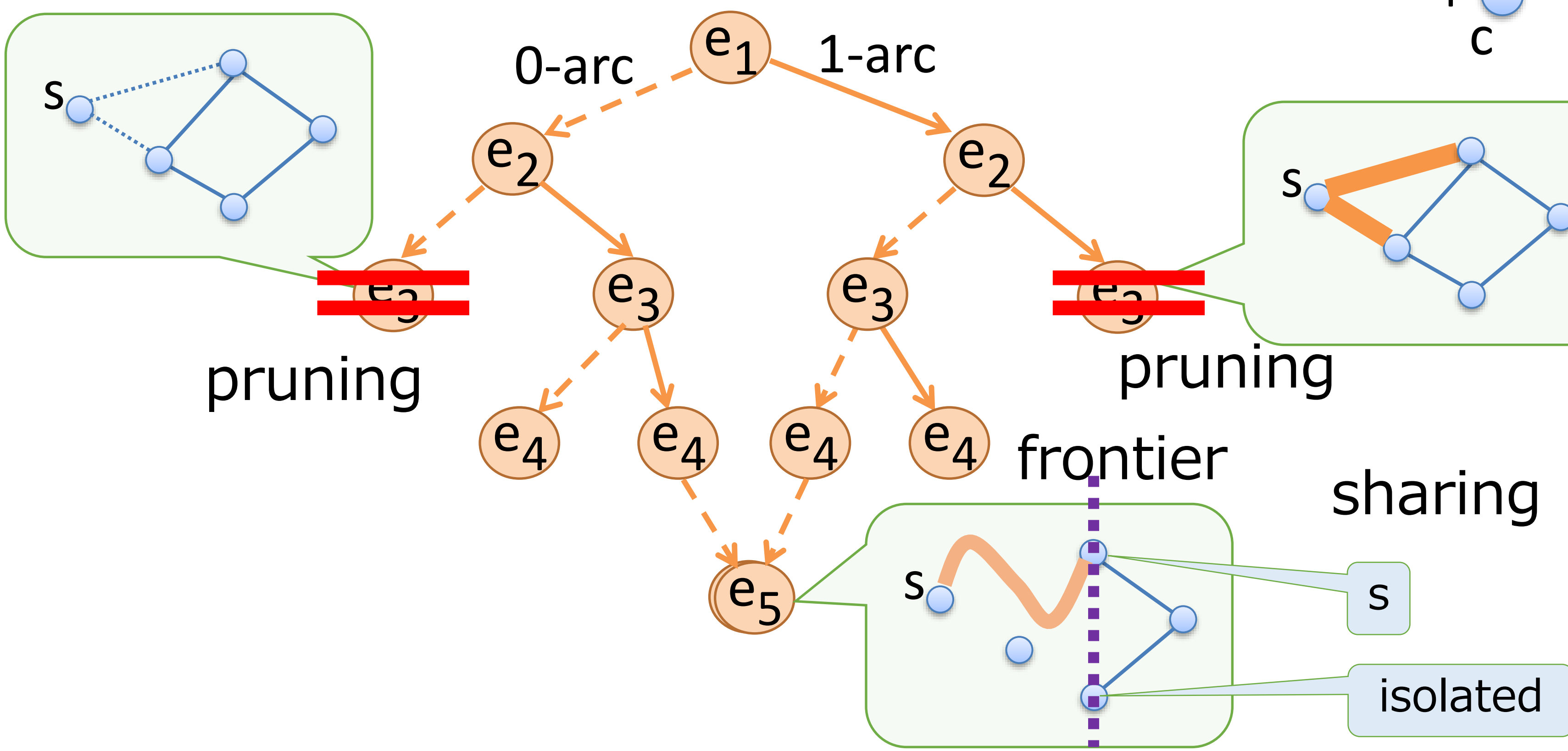
ZDD construction: frontier-based search (FBS)

[Sekine et al. 1995], [Knuth 2008], [Kawahara et al. 2017]

- constructs the ZDD representing a set of subgraphs (e.g., s-t paths)
  - in a top-down manner
  - by pruning and sharing nodes



ex.) s-t paths

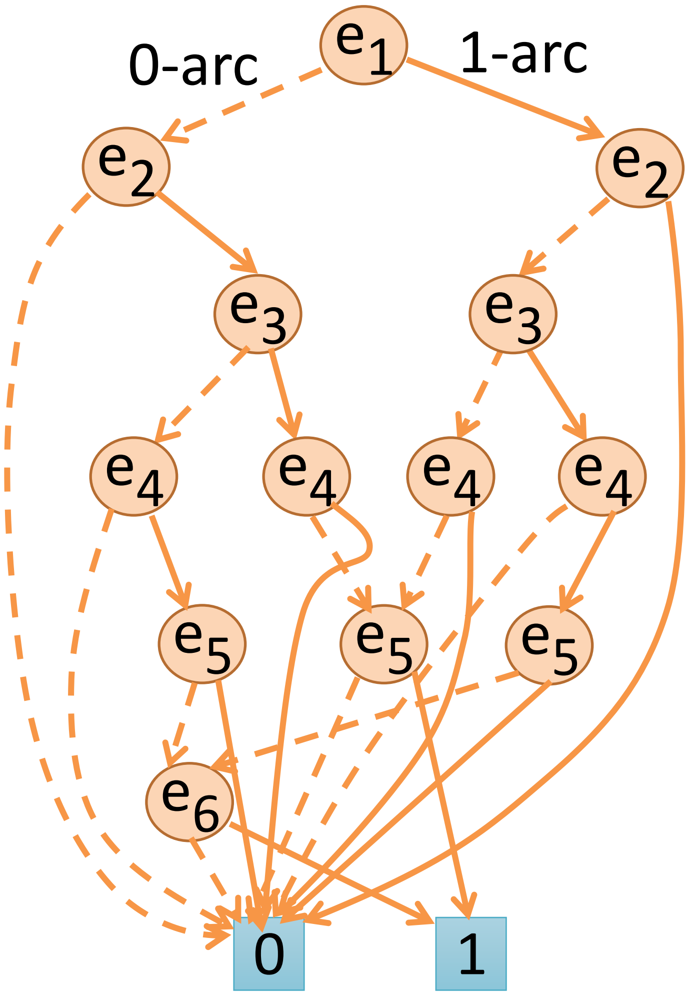


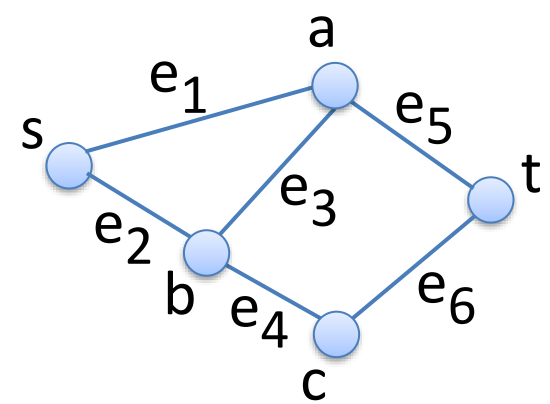
ZDD construction: frontier-based search (FBS)

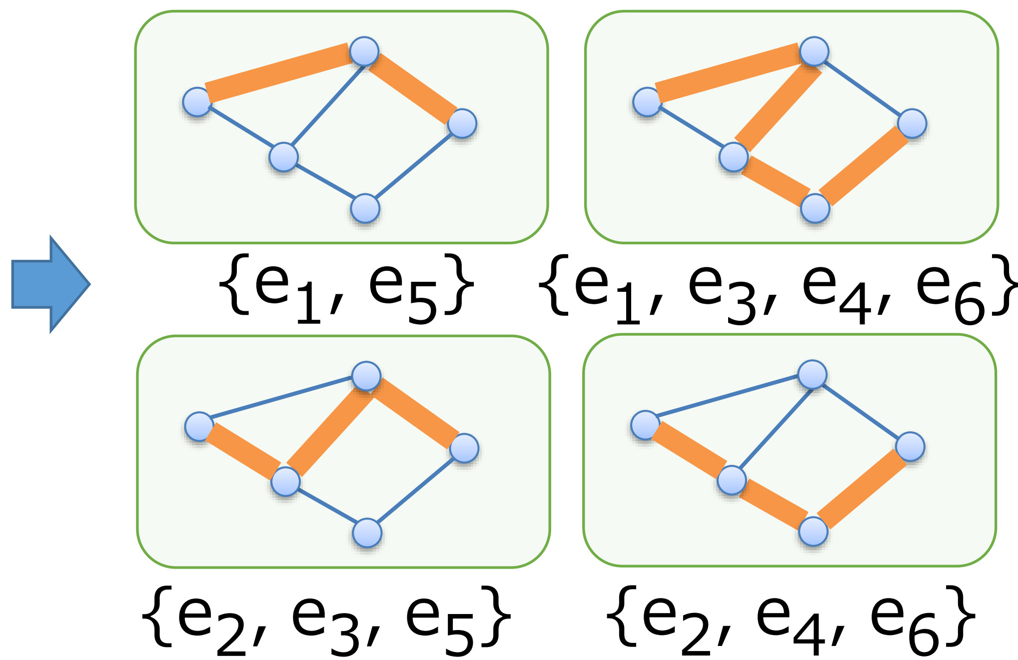
[Sekine et al. 1995], [Knuth 2008], [Kawahara et al. 2017]

- constructs the ZDD in a top-down manner

resulting ZDD







Families of subgraphs

- We can construct ZDDs for various kinds of subgraphs.

Subgraphs treated by [Sekine+ 1995][Knuth 2011][Kawahara+ 2017]

s-t paths
cycles
trees, forests
spanning trees
Steiner trees
matchings
degree constrained graph
...

The ZDD solver can solve the above subgraph reconfiguration problems.

Characterized by forbidden subgraphs

[Kawahara+ 2019]

chordal graphs
interval graphs
proper interval graphs
...

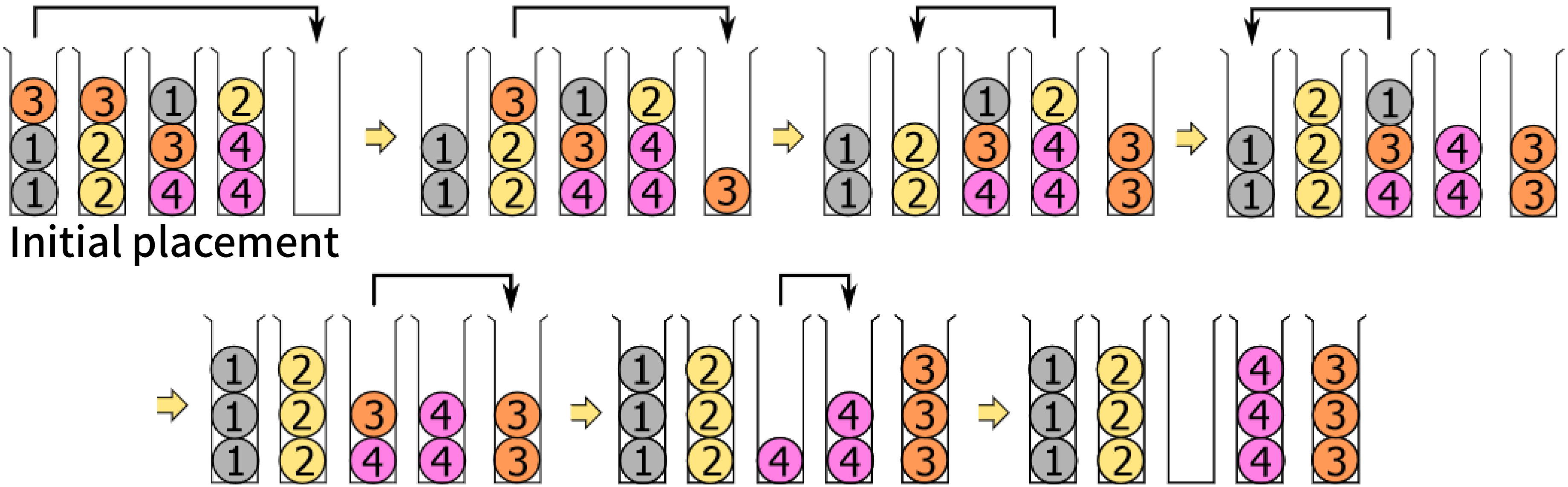
Characterized by forbidden minors

[Nakahata+ 2020]

planar graphs
outer planar graphs
series parallel graphs
cactus
...

Application: Ball sort puzzle

We can move a ball to an empty bin or on a ball with the same color.



The numbers of balls represent just colors.

Goal:
All the balls in each bin have the same color.

arXiv > cs > arXiv:2202.09495

Search...

Help | Advance

Computer Science > Computational Complexity

[Submitted on 19 Feb 2022]

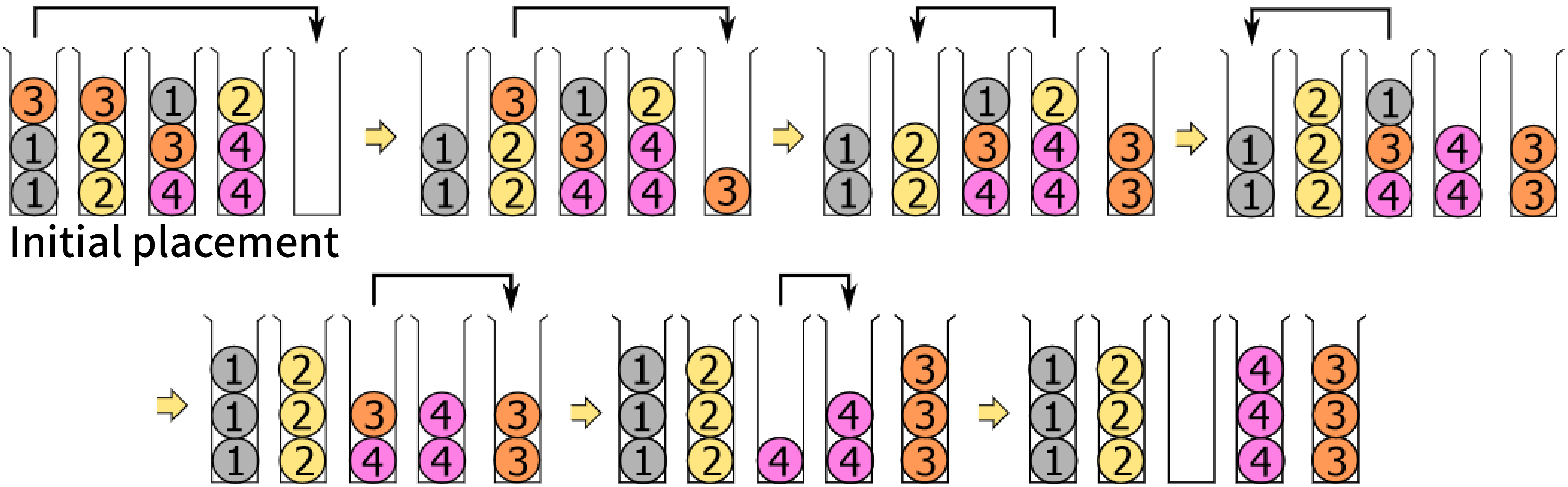
Sorting Balls and Water: Equivalence and Computational Complexity

Takehiro Ito, Jun Kawahara, Shin-ichi Minato, Yota Otachi, Toshiki Saitoh, Akira Suzuki, Ryuhei Uehara, Takeaki Uno, Katsuhisa Yamanaka, Ryo Yoshinaka

To appear in FUN 2022

Application: Ball sort puzzle

We can move a ball to an empty bin or on a ball with the same color.



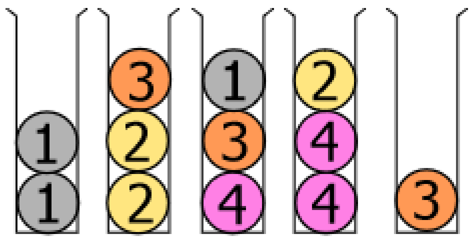
The numbers of balls represent just colors.

Goal:
All the balls in each bin have the same color.

Complexity [Ito+ 2022]
Deciding whether a reconf sequence exists or not: NP-complete.
Polynomial solvable if the capacity of bins $h = 2$, the number of colors is n.
Relation between the number of empty bins and solvability.

Construction of solution space ZDD

We represent the placement of balls as a set.



$\{v_{111}, v_{121}, v_{212}, v_{222}, v_{233}, v_{314}, v_{323}, v_{331}, v_{414}, v_{424}, v_{432}, v_{513}\}$

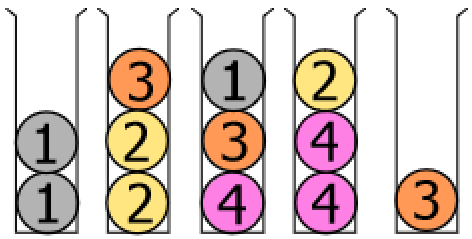
$v_{i,j,c}$ There is a ball with color k at the j-th position (from the bottom) in the i-th bin.

Let $\mathcal{X}_{i,j,c}$ be the family of all the sets including $v_{i,j,c}$.
Let $\overline{\mathcal{X}_{i,j,c}}$ be the family of all the sets not including $v_{i,j,c}$.

$$\mathcal{X}_{i,j,c} = \{ \{v_{i,j,c}\} \cup X \mid X \subseteq U \setminus \{v_{i,j,c}\} \}$$
$$\overline{\mathcal{X}_{i,j,c}} = 2^U - \mathcal{X}_{i,j,c}$$

Construction of solution space ZDD

We represent the placement of balls as a set.



$\{v_{111}, v_{121}, v_{212}, v_{222}, v_{233}, v_{314}, v_{323}, v_{331}, v_{414}, v_{424}, v_{432}, v_{513}\}$

$v_{i,j,c}$ There is a ball with color k at the j-th position (from the bottom) in the i-th bin.

Let $\mathcal{X}_{i,j,c}$ be the family of all the sets including $v_{i,j,c}$.
Let $\overline{\mathcal{X}_{i,j,c}}$ be the family of all the sets not including $v_{i,j,c}$.

$$\mathcal{X}_{i,j,c} = \{\{v_{i,j,c}\} \cup X \mid X \subseteq U \setminus \{v_{i,j,c}\}\}$$
$$\overline{\mathcal{X}_{i,j,c}} = 2^U - \mathcal{X}_{i,j,c}$$

(i) At most one ball exists at the same place.

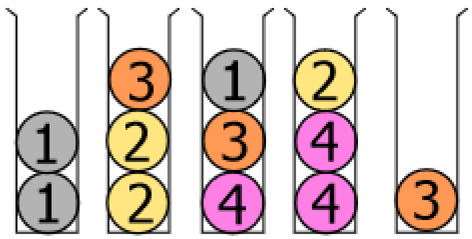
$$\bigcap_{i,j} \left\{ \begin{array}{l} \text{set obtained by} \\ \text{choosing at most one of } v_{i,j,1}, v_{i,j,2}, \cdots \\ \text{(and other elements arbitrarily)} \end{array} \right\}$$

(ii) If a ball exists at position j (≥ 2), a ball exists at position $j-1$.

$$\bigcap_{j\geq 2,i} \bigcup_c \mathcal{X}_{i,j,c} \Rightarrow \bigcup_c \mathcal{X}_{i,j-1,c}$$

Construction of solution space ZDD

We represent the placement of balls as a set.



$$\{v_{111}, v_{121}, v_{212}, v_{222}, v_{233}, v_{314}, v_{323}, v_{331}, v_{414}, v_{424}, v_{432}, v_{513}\}$$

$v_{i,j,c}$ There is a ball with color k at the j-th position (from the bottom) in the i-th bin.

Let $\mathcal{X}_{i,j,c}$ be the family of all the sets including $v_{i,j,c}$.
Let $\overline{\mathcal{X}_{i,j,c}}$ be the family of all the sets not including $v_{i,j,c}$.

$$\mathcal{X}_{i,j,c} = \{ \{v_{i,j,c}\} \cup X \mid X \subseteq U \setminus \{v_{i,j,c}\} \}$$
$$\overline{\mathcal{X}_{i,j,c}} = 2^U - \mathcal{X}_{i,j,c}$$

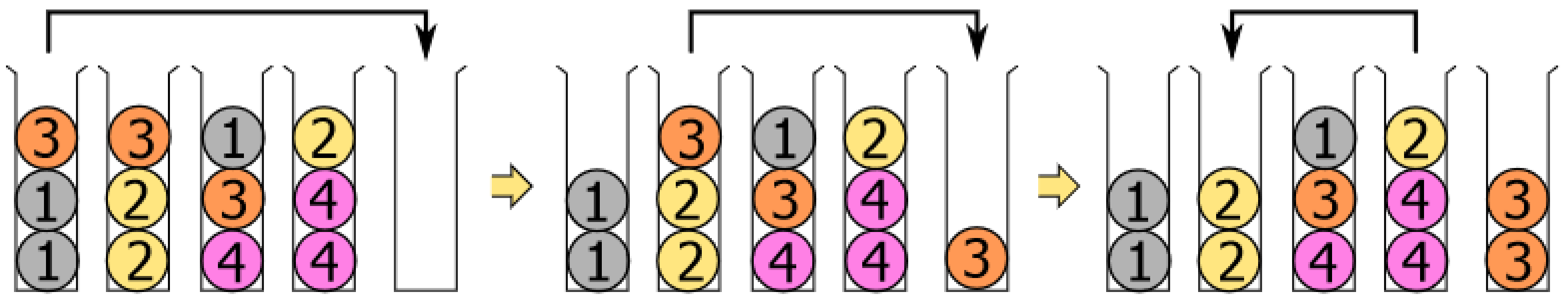
(iii) There are exactly h balls with the same color.

$$\bigcap_c \left\{ \begin{array}{l} \text{set obtained by choosing exactly } h \text{ of } v_{1,1,c}, v_{1,2,c}, \cdots \; v_{2,1,c}, v_{2,2,c}, \cdots \\ \text{(and other elements arbitrarily)} \end{array} \right\}$$

By taking the intersection of (i), (ii), (iii), we obtain the solution space ZDD.

One-way ball move operation

We can move a ball to an empty bin or on a ball with the same color.



This restriction of the movement cannot be represented by imposing the solution space (ZDD).

The ball placement is still valid even if we move a ball on another ball with a different color.

ZDD operation of the one-way ball move

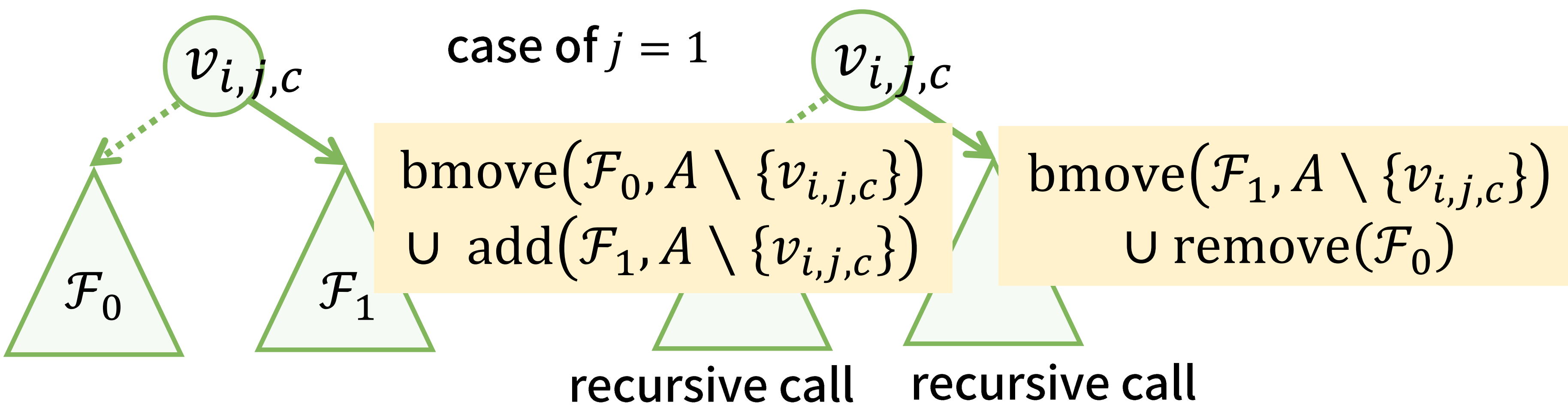
Determine the order of ZDD variables so that $v_{i,j,c} < v_{i,j-1,c}$.

(The order of other variables is arbitrary.)

$$\mathrm{bmove}(\mathcal{F}, A) = \{\, F \cup \{v\} \setminus \{v'\} \mid F \in \mathcal{F}, v \notin F, v \in A, v' \in F\}$$

and satisfying the move condition

We consider only the case where $v_{i,j,c}$ is the smallest in A.



same as swap

ZDD operation of the one-way ball move

Determine the order of ZDD variables so that $v_{i,j,c} < v_{i,j-1,c}$.

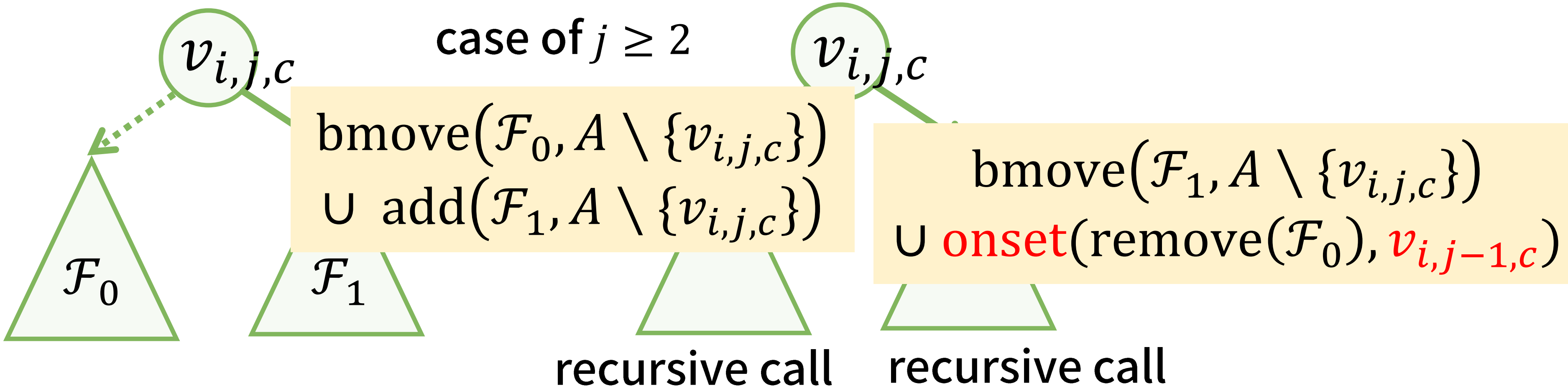
(The order of other variables is arbitrary.)

$$\mathrm{bmove}(\mathcal{F}, A) = \{\, F \cup \{v\} \setminus \{v'\} \mid F \in \mathcal{F}, v \notin F, v \in A, v' \in F\}$$

and satisfying the move condition

We consider only the case where $v_{i,j,c}$ is the smallest in A.

case of $j \geq 2$



$$\mathrm{onset}(\mathcal{F}, v) = \{\, F \in \mathcal{F} \mid v \in F\}$$

Ball sort puzzle solver

- Let $\mathcal{F}_{\mathrm{sol}}$ be the set of all the feasible placements.
- Let S be an initial placement.
- $\mathcal{F}_0 \leftarrow \{\, S \,\},\ i \leftarrow 1$
- $\mathcal{F}_i \leftarrow \mathrm{bmove}(\mathcal{F}_{i-1}, V) \cap \mathcal{F}_{\mathrm{sol}}$

We need not remove past sets.

all the sets obtained by one step

extract only feasible placements

- If $\mathcal{F}_i$ is empty, output "no reconf sequence"
- If $\mathcal{F}_i \cap \mathcal{F}_{\mathrm{goal}} \neq \emptyset$, output "There is a reconf sequence from S to a goal with length i."

(ZDD for) the family of all the goal placements.

- $i \leftarrow i + 1$, and continue.

The reverse operation of bmove

- We can consider the reverse operation of bmove.
- To perform the reverse operation to $\mathcal{F}_{\mathrm{goal}}$ repeatedly, we can construct the ZDD representing the family of all the placements.
    - It enables to enumerate the instances of the puzzle.
    - We can obtain an instance with the longest sequence.
    - Using a feature of ZDDs, we can sample an instance uniformly at random.

Python interface

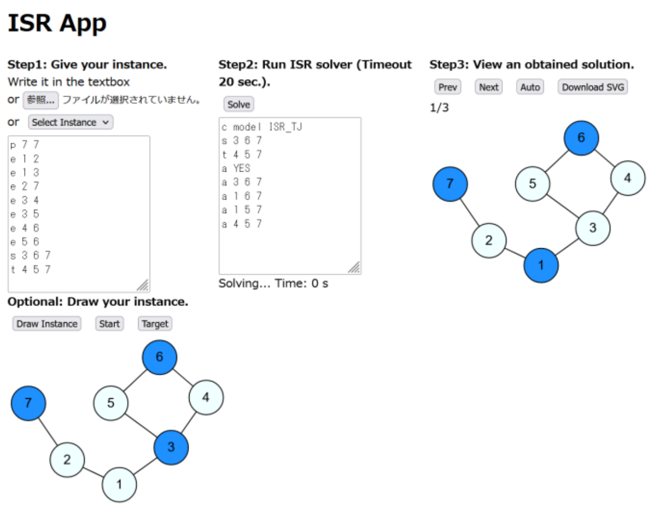
- We are developing a Python interface for reconfiguration problems.

```
# 3 x 3 grid
vertices = [1, 2, 3, 4, 5, 6, 7, 8, 9]
edges = [(1, 2), (1, 4), (2, 3), (2, 5),
         (3, 6), (4, 5), (4, 7), (5, 6),
         (5, 8), (6, 9), (7, 8), (8, 9)]
# Sets ZDD variables
setset.set_universe(vertices)

# Computes (the ZDD for) all the independent sets
iss = reconf.get_independent_setset(vertices, edges)

s = {2, 4, 6} # initial set
t = {1, 6, 8} # goal set

# Obtains a reconf sequence (support TJ, TS, TAR)
seq = reconf.get_reconf_seq(s, t, iss)

for x in seq:
    print(x)
```

will be published soon...

SAT-based solver

- SAT-based solver
  - You can use the SAT-based solver from Web (but large graph cannot be solved).

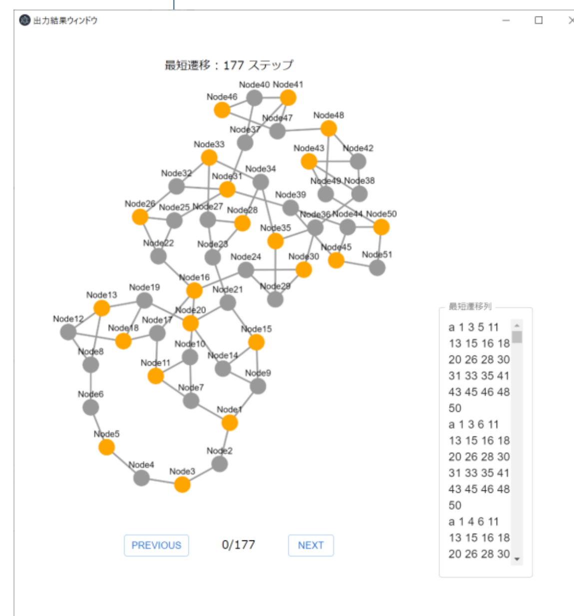


https://israpp.herokuapp.com/

Software with GUI

- We are also developing software with GUI.

(support Windows/Mac/Linux)



will be published soon…

Conclusion

The paper will be published soon.

- We introduced ZDD representing the family of sets.
  - Compressed representation
  - Rich set operations
- We proposed a ZDD-based solver for various reconfiguration problems.
  - Constructing the ZDD for the set of feasible solutions.
  - Desinging one-step operations such as remove, add, swap, bmove,...
  - breadth-first search
- The analysis of the algorithm is proceeding...
  - We obtain some results, but we don't introduce it.
- It seems difficult to improve ZDD operations.

  quite simple
- Further experiments are needed.